REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Aug 96	

4. TITLE AND SUBTITLE
Simulation of Droplet Evaporation In Supercritical Environments Using Parallel Molecular Dynamics

5. FUNDING NUMBERS

6. AUTHOR(S)
Jeff ery K. Little

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Pennsylvania State University

8. PERFORMING ORGANIZATION REPORT NUMBER
96-061

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
DEPARTMENT OF THE AIR FORCE
AFIT/CI
2950 P STEET, BLDG 125
WRIGHT-PATTERSON AFB OH 45433-7765

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

14. SUBJECT TERMS

15. NUMBER OF PAGES
154

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

The Pennsylvania State University

The Graduate School

Department of Aerospace Engineering

SIMULATION OF DROPLET EVAPORATION IN SUPERCRITICAL ENVIRONMENTS USING PARALLEL MOLECULAR DYNAMICS

A Thesis in

Aerospace Engineering

by

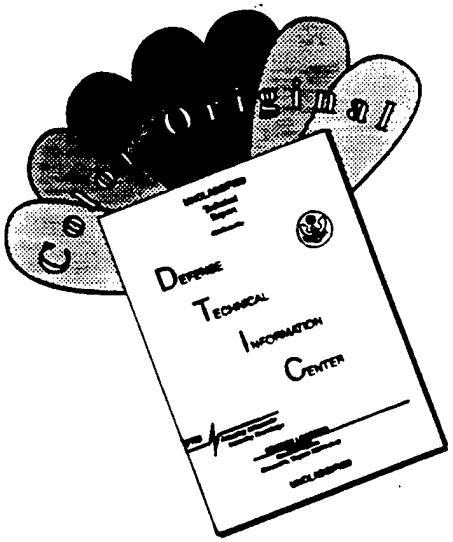
Jeffery K. Little

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 1996

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

The Pennsylvania State University

The Graduate School

Department of Aerospace Engineering

SIMULATION OF DROPLET EVAPORATION IN SUPERCRITICAL ENVIRONMENTS USING PARALLEL MOLECULAR DYNAMICS

A Thesis in

Aerospace Engineering

by

Jeffery K. Little

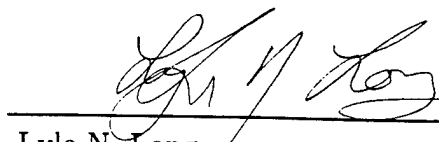
Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 1996

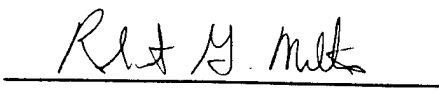
We approve the thesis of Jeffery K. Little.

Date of Signature



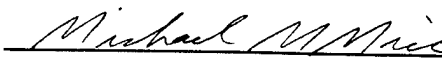
July 15, 1996

Lyle N. Long
Associate Professor of Aerospace Engineering
Thesis Adviser
Chair of Committee



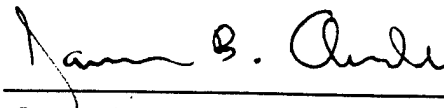
July 16, 1996

Robert G. Melton
Associate Professor of Aerospace Engineering



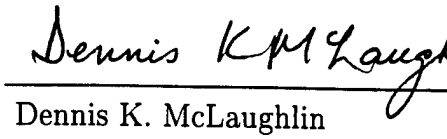
7/16/96

Michael M. Micci
Associate Professor of Aerospace Engineering



July 15, 1996

James B. Anderson
Evan Pugh Professor of Chemistry



July 17, 1996

Dennis K. McLaughlin
Professor of Aerospace Engineering
Head of the Department of Aerospace Engineering

ABSTRACT

The complete evaporation of three-dimensional submicron droplets under both subcritical and supercritical conditions has been modeled using molecular dynamics (MD). This work represents a first step toward an accurate analytical modeling of combustion in supercritical environments. In this initial study the two-phase simulations consist entirely of argon atoms distributed between a single droplet and its surrounding vapor. The inter-atomic forces are based on a Lennard-Jones 12-6 potential, and the resultant atomic displacements are determined using a modified velocity Verlet algorithm. Linked cell lists in combination with Verlet neighbor lists allow efficient modeling of the large and diverse simulations. A non-cubic periodic boundary, specifically a truncated octahedron, is used to minimize periodicity effects. A unique method, using the linked cell structure, streamlines the associated boundary computations. The linked cells are also used as domains for density, temperature and surface tension computations. This allows a contouring of these properties. The surface tension measure is a unique development.

All of these techniques are incorporated into a message passing code for use on the IBM SP2 parallel computing platform. A particle decomposition technique successfully provides nearly perfect load balancing across any desired number of processors. The resultant code was compared to the best known timings of other MD models on single and multiple processor computers. It proved quite capable when running a bulk liquid benchmark simulation. Only 24 processors were required to surpass the best vectorized serial performer. The capability on 32 processors of the SP2 also approached 50% of the speed documented for similar simulations on 256 processors of the Cray T3D and 512 processors of the Intel Paragon parallel machines.

Six simulations were performed as part of this work. The first four involved the evaporation of a 5,600 atom drop suddenly exposed to assorted surroundings. The final two are duplicates of the fourth, a supercritical case, but with 27,000 atoms and 100,000 atoms respectively in the initial drop. These were used for scaling studies. Contour property imaging was performed for each case, and evaporation rates were

measured using an area factor based on local density levels. Each system was initialized by independently modeling the environment and the drop and then fusing them together. The fuse technique uses vector algebra to allow a smooth combining of any droplet shape into an equilibrated environment. Additionally it ensures the retention of the equilibrated surface energy. This is believed to be a unique approach.

The first two of the small drop diffusions were subcritical runs. They provide validating information and insight into the diffusion process. The first, at an environment state point close to the vapor curve, reveals a slow quasi-equilibrium process due to the relatively low thermal and density gradients. The second environment was at an elevated temperature but subcritical pressure. The resultant low density allowed the droplet to retain a strong surface energy, but the large thermal gradients enhanced the evaporation rate. A D^2 evaporation analysis was performed with reasonably good agreement obtained (within 20%).

The next two cases, a near-critical and a supercritical simulation, both provide interesting results for review and analysis. The near-critical case was set at the critical density and just over the critical temperature. The high density caused the surface energy to dissipate immediately. Low thermal gradients, though, significantly reduced the surface diffusion. In fact, the contour images imply that a cloud-like mixing overshadowed the diffusion rate as a result. The supercritical case is very interesting. At a state point of 1.3 T_c and 1.5 P_c, where the c subscript denotes the critical state, the surface tension still dissipated quickly. A diffusion layer at the critical temperature was established, and a non-spherical evaporation of the droplet ensued. This case was also the basis for the supercritical scaling study.

The successful scaling of this run and the 27,000 atom droplet diffusion to the large simulation of the 100,000 atom drop at the same conditions verified that the supercritical diffusion operates as a purely surface effect. In other words, the three cases revealed equivalent evaporation rates occurring at similarly regressing surface profiles. This implies that the microscopic evaluations can be extended and applied to macroscopic applications.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I express my sincere appreciation to those who have made this accomplishment possible. This work is far from an individual effort.

First, I would like to thank Col. Michael Smith, whose confidence and recommendation led directly to the sponsoring of my work. I also gratefully acknowledge the Department of Aeronautics and Astronautics at the Air Force Institute of Technology for this sponsorship.

There are numerous individuals who generously provided their time and knowledge over the course of this endeavor. The fellow members of the molecular dynamics research group, Teresa Kaltz, Obika Nwobi, Chad Ohlandt and Brian Wong, have all contributed numerous hours of productive discussions. The Numerical Intensive Computing group, notably Kevin Morooney and Jeff Nucciarone, have established an exceptional computing platform in the IBM SP2. Their ready support and innovative management were very helpful. Also my appreciation is extended to my officemates, Drs. Vineet Ahuja, Thomas Chyczewski, Dale Hudson, Yusuf Özyörük, and Zvi Weinberg. They helped considerably.

I also appreciate the support from the members of my thesis committee. My thesis advisor, Dr. Lyle Long, provided exceptional guidance. His generous support, insightful critiques, and ready encouragement were instrumental in the production of this thesis. My thanks are also extended to the other members, Drs. Robert Melton, Michael Micci and James Anderson, for their thorough review of this thesis and perceptive comments. Additionally, Dr. Micci's insights and challenges throughout the past two years are readily apparent in this work.

Finally I would like to thank my family. To my father and mother, thank you for instilling in me a much appreciated mix of drive and patience. To my brother, my first tutor, you helped more than you realize. To my girls, Sarah, Kelly and Samantha, your patience and energy are inspirational. And finally, to my wife, Ellen, thank you for your unflinching support. You, more than anyone, understands how truly sincere are these words of appreciation. I am very blessed.

Chapter 1

INTRODUCTION

Diffusion flames are found in many modern combustion devices driving numerous aerospace vehicles. Characterized by initially separate fuel and oxidizer regions, the rate of this combustion is driven by the mixing rate. In fact the chemical reaction rate can often be neglected entirely by comparison [1]. When a fuel droplet is exposed to an oxidizer environment at low temperatures and pressures, this mixing rate is very slow. Above the thermodynamic state termed the critical point, however, the fuel and oxidizer combine very rapidly. Rapid mixing results in cleaner, more stable, and more complete combustion, primary goals of any combustor design. So a thorough understanding of supercritical combustion is very desirable.

The supercritical region, depicted in figure 1.1 [2] [3], is defined as any state above the critical temperature and pressure [4]. Under these conditions a fluid can not be compressed into a liquid. Due to the lessening of the demarcation between liquid and gas phase densities, a fuel droplet exposed to this environment will experience a significant decrease in its surface energy. Also, the high temperature oxidizer consists of energized atoms moving at high velocities. The combination of the resultant high speed collisions and the lower surface energy generates the desirable rapid mixing.

Current rocket motors, gas turbines, diesel engines, and many projected advanced combustor designs operate supercritically. While concepts are well established from theory and experiment for subcritical conditions, there are no acceptable theories to fully describe the physics at and above the critical point [5]. This leaves many questions unanswered in the minds of combustion engineers as they attempt to design efficient mixing of subcritical fluid droplets into supercritical environments. The retention of a spherical structure is often assumed for the droplets during combustion. The reduction of the surface tension when exposed to supercritical conditions, however, places this assumption in question. Under the influence of such a high pressure and temperature environment, one might also question the approach of defining the

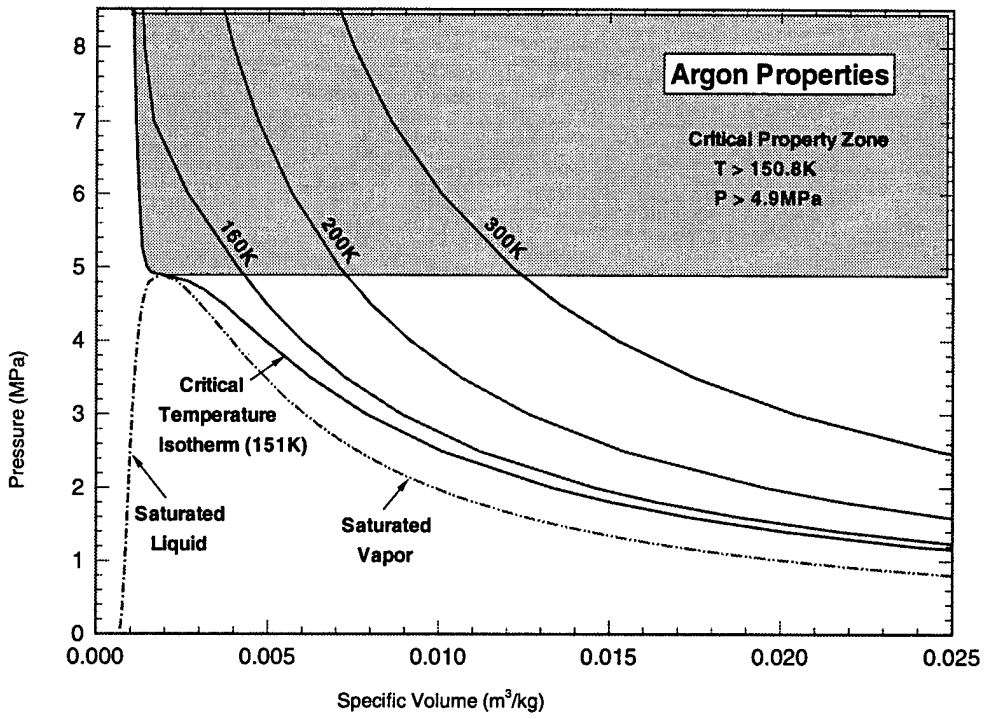


Figure 1.1. Argon Pv diagram. The shaded area represents the supercritical region.

diffusion as a surface phenomena. Instead the droplet may change to a supercritical phase within its interior prior to the completion of diffusion. To help clarify such design considerations, a code to model on the molecular level the diffusion process of saturated liquid droplets in supercritical environments was developed. The results of initial investigations utilizing the code are detailed in this paper.

1.1 Problem Definition

Simply stated, an analytical means of studying the diffusion process which occurs when a fuel droplet is exposed to a supercritical environment is desired. Experimental investigations of such a dynamic process in high temperatures and pressures are difficult. Classic analytical techniques, such as continuum-based computational fluid dynamics, require far reaching assumptions and depend on suspect property

data. The path chosen to achieve the desired analysis therefore involves molecular dynamics algorithms. The reason for this choice will soon become apparent.

Molecular dynamics (MD) is a term used to describe the modeling of individual molecular motions due to intermolecular forces. The evaluation of aggregate and individual particle positions and velocities allows the computation of macroscopic properties. Comparisons of MD-generated property values to those based on established theory are used for code verification, but MD has much greater utility. Intermolecular forces in soft sphere MD models (the type used in this work) are computed based on assumed intermolecular potential functions. The intermolecular potentials are based on the atomic interactions of unbalanced electron charges [6]. They are microscopic interactions which exist regardless of the macroscopic state. Therefore, an MD code can be used to model situations where macroscopic properties are, to date, undefined or, at best, ill-defined. One such case is the modeling of supercritical phase changes.

Traditionally, the phases of matter are thought of as distinctly different in form and properties. The solid phase consists of molecules which vibrate about fixed locations. The atomic positions are still based on the potential forces previously mentioned, but the atoms have settled into a lattice structure. They remain in relative fixed positions because they do not have enough energy to break away from their neighbors. Gaseous phase molecules, by contrast, display no order. They have high energy levels and move in and out of the potential fields like billiard balls. Liquids, as expected, represent a transition phase. Many of the atoms are clustered together in solid-like structures, but these clusters move quite freely; they have broken away from the solid lattice (melted). This breaking away is a result of atoms occasionally reaching speeds high enough to break through the potential well. Figure 1.2 depicts these concepts and shows in the radial distributions that liquids have short-range order, while solids are infinitely ordered and gases completely random [7] [6].

Above the critical point, however, every atom has enough energy to break the potential field. No matter how close the molecules are packed, they will continue to exhibit the gas-like interaction of breaking through the potential well. The molecules are also very tightly spaced, however, and will continuously be affected by neighbor potentials. This combination of gas and liquid related interactions is what makes

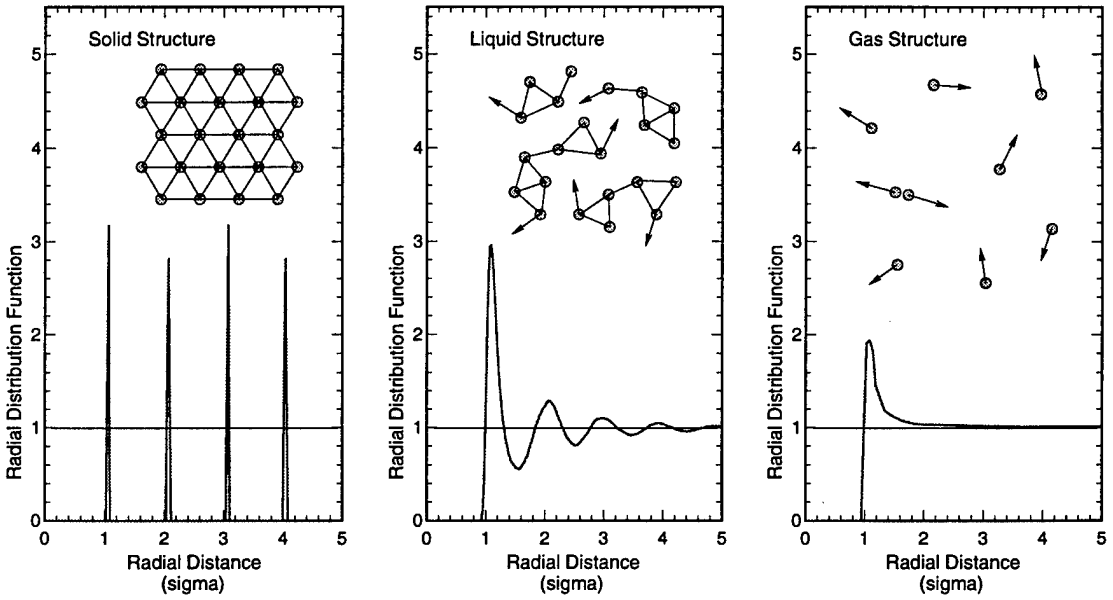


Figure 1.2. Classic phases of matter. The images are for demonstration only. Actual solid and liquid structures are more closely packed in 3-D structures.

evaluating the supercritical region so difficult. Molecular dynamic modeling effectively eliminates the problems by directly incorporating the potential field interactions. Analytical evaluations are therefore possible. There is one limitation for such studies, however: system size.

As mentioned, subcritical liquids display short-range order while gases show no order at all. One might suspect that supercritical liquids would demonstrate very short-range order. Unfortunately, this is not the case. Instead, the closely packed molecules demonstrate a long order (evidenced in a radial distribution plot by a wide single peak). At the critical point this order approaches infinite proportions. So modeling phase transitions through the critical region appears beyond reach. Cummings [8] however, notes that such simulations are possible due to the dominance of short-range solubilites compared to the long-range critical order. Care must still be taken to ensure large enough systems are modeled to include the complete structural orders in the simulation domain. Further details concerning this are provided in the literature review section of this chapter.

Until recently, computer capabilities limited molecular dynamic models to systems on the order of a few thousand particles. Existing massively parallel architectures can model millions of particles. The continued improvements in parallel computing platforms will allow sophisticated analytical modeling of the supercritical phase. This project, for example, involves systems large enough to perform scaling studies. The scaling of small simulation results to represent large physical systems is the necessary link to ultimately allow the modeling of actual combustor designs. There are possible drawbacks associated with parallel computing, however. Unfortunately, molecular dynamics codes are especially susceptible.

Parallel computers are essentially many processors linked together and set to operate concurrently. If a given computational task is shared evenly among the processors, the task can be efficiently accomplished. This job sharing is analogous to two men painting a room. Assuming they are equal in ability, they will complete the project the quickest if they have an equal number of walls to paint. Not all projects are so easily divided, however. Suppose the same two men are writing a book together. They cannot simply perform their tasks independently; they must communicate information back and forth to allow the book to make sense. Suppose ten people are trying to collaborate on the writing of a book. The communications could be so great as to significantly increase the complexity of the task. Many workers could take longer than a few. Programming parallel architectures requires similar considerations. Even division of the computational workload and minimal communication between processors are both important factors in generating efficient parallel code.

The even division of computational work is termed load balancing. Molecular dynamic codes are especially sensitive to load imbalances. Finite difference grids will see a linear increase in computations due to the density of the grid network. For a given geometric space, the number of grid points is directly proportional to density. Molecular dynamic systems, however, have a workload which is proportional to the square of the density. Not only are there more atoms to evaluate, there are also more neighboring atoms contributing to the inter-atomic forces. Since we are modeling a

dense mass (a liquid) in a lower density environment (even critical point densities are less than half the liquid densities) load balancing is an important concern.

Careful coding of parallel molecular dynamics was therefore the approach chosen for the model detailed herein. This tool was envisioned to provide an informative and physically accurate simulation of a droplet diffusing into a supercritical environment. The parallel architecture would also allow larger evaluations for scaling studies. Before detailing the work further, a brief review of related work is provided.

1.2 Literature Review

At this point the reader should appreciate that the work detailed herein is multifaceted. It involves the combining of combustion related fluid dynamics, molecular dynamics, and the high speed capabilities of parallel computers. Molecular dynamics, the study of substances based on modeling molecular displacements and interactions, has traditionally been utilized by chemists. Today this relatively new science is gaining ground in a wide variety of disciplines, including aerodynamics. The full background review for this thesis, therefore covers a wide array of topics.

1.2.1 Combustion

In 1928, Burke and Schumann [9] first introduced the concept of diffusion flame theoretical modeling. Their theory was very useful, although somewhat flawed, but did not apply to high pressure environments. Spalding [10], in 1959, and later Rosner [11], in 1967, addressed theoretical errors associated with many of the modeling assumptions when applied to high pressure combustion. These errors mainly stemmed from the erroneous assumption of quasi-steady combustion in this environment. Savery and Borman [12] showed experimentally while Manrique and Borman [13] detailed numerically that steady state conditions cannot be met at the critical point. In effect, these early researchers detailed the breakdown in combustion theory for critical and supercritical environments. This lack of knowledge continues today.

Yang et al. [14] provide a very impressive attempt at filling the void of information concerning supercritical diffusion in their 1994 paper. The authors describe

a detailed analysis of the vaporization of liquid oxygen droplets into supercritical hydrogen environments. Their work is based on continuum physics and the solution of a complete set of conservation equations. This is essentially analogous to the development of the classic D^2 law which successfully models subcritical droplet evaporation. The strength of their research, however, rests on using an advanced equation of state which allows modeling from a compressed liquid to a dilute gas with a single property evaluation scheme. When supercritical diffusion begins in their model, a single phase analysis is used. The droplet surface is then defined as the point where the critical mixing temperature occurs.

This impressive work is not, in the opinion of this author however, an accurate description of the diffusion dynamics. The assumption of a finite spherical control volume where thermodynamic phase equilibrium is maintained for the life of the process is a questionable premise. Subcritical droplets exist as nearly perfect spheres because they are minimizing the relatively large surface tension present. The tension exists because of the significant density variations across the droplet and surroundings interface. In supercritical diffusion this surface energy is very small; the droplet cannot be assumed to remain a sphere. Also, the authors admit that much of the property estimations they are using would be better met by rigorous intermolecular model studies. They used a more heuristic approach since the rigorous studies did not yet exist. As a result, future collaborative work between this paper's molecular dynamic research and Yang's work is very likely.

As a close to this section, a recent review of top combustion engineers revealed the import they hold for attaining knowledge concerning supercritical effects. In the 25th anniversary edition of *Combustion Science and Technology*, Glassman asked experts in the field to predict future challenges and successes. Williams [15], Smyth [16], and Brezenski [17] all mention the requirement for significant advances in near-critical and supercritical theories. Williams includes the topic in his 'top ten' list of significant developments for the next 25 years. Smyth calls for a joining together of chemist and fluid mechanicians, and Brezenski calls 'further, enduring research' into supercritical thermodynamics a 'timely necessity.'

1.2.2 Molecular Dynamics

Evans and Hoover [18] highlight in a 1986 article that Fermi began Molecular Dynamics studies in 1955 with 16 particles, but by 1984, a 'world record' simulation of 161,604 particles was performed. Just ten years later, Beazley and Lomdahl [19] modeled 600 million particles (but for only a few time steps). While this sounds very impressive, the time to run such a simulation is still extraordinary. Large systems are on the horizon, but many important smaller studies should not be overlooked.

Molecular dynamics has been used to support chemistry and biology research for many years. In 1994, Cummings [8] discussed the work he was aware of in the fields of near-critical and supercritical MD modeling of fluids. All of his references involved dilution studies of a small number of solutes in a relatively large near-critical solvent. The solute models were on the order of single particles while the solvents were modeled with approximately a thousand. The solvent populations were large to appropriately model the long range effects present at the critical point.

Cummings recommended using model sizes at least twice the size of the correlation length, the length of significance on the radial distribution plot (see figure 1.3). To estimate this size he also recommended Sengers and Levelt Sengers [20] approximation of

$$\xi = \xi_0 \mid \Delta T^{\star} \mid^{-\nu} \tag{1.1}$$

where the dimensionless parameter, $\Delta T^{\star}$, is defined as

$$\Delta T^{\star} = \frac{T}{T_c} - 1$$

and for Argon

$$\xi_0 = 0.14 \times 10^{-9} meters$$

$$\nu = 0.63$$

This formulation is based on the theory of universality (all substances behave identically at the critical point), and the assumption that along a critical isochore, $\rho = \rho_c$, power law expansions of thermodynamic properties can be set as functions of $\Delta T^{\star}$.

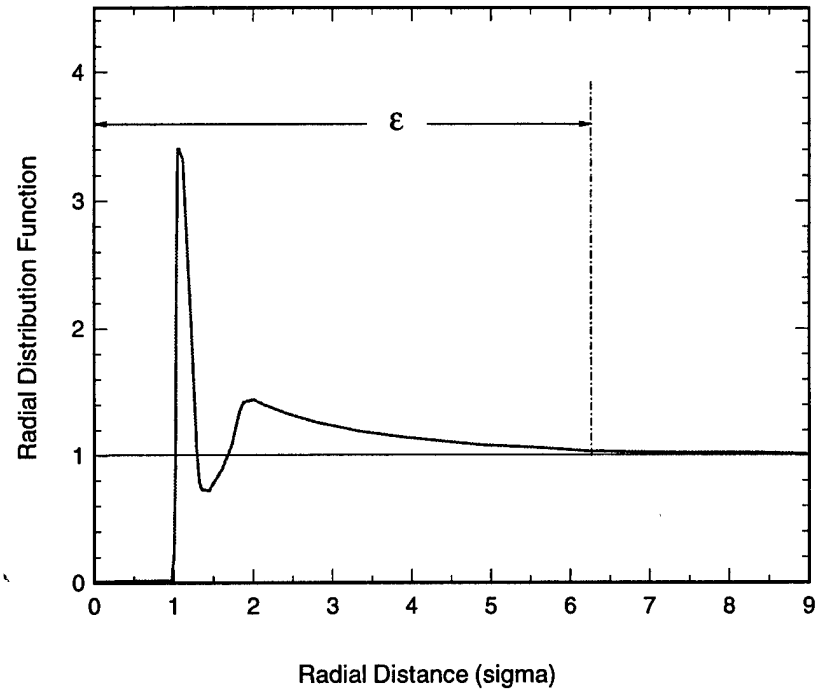


Figure 1.3. Correlation length. The correlation length is the distance to the point where the structure shows no order. This plot is indicative of the long orders of near-critical structures [20].

The term ν is a critical exponent of the power law expansion while ξ_0 is the amplitude of the asymptotic power law for ξ. Equation 1.1 is considered valid only for $0.9 < \frac{T}{T_c} < 1.0$.

Note, the correlation length, ξ, is undefined at the critical temperature. This seems to imply that going through this point would require infinite MD model sizes. But Cummings notes that his own work on aqueous systems have shown that these long-range effects due to nearness to criticality are secondary to the short-range solubilities. So while the long correlation lengths near the critical point require careful modeling of sufficiently large environments, they do not negate the utility of molecular dynamics for this study.

Chemists have also simulated droplets through MD. Most notably, Thompson et al.[21] have investigated droplet curvature effects at both Cornell and Oxford.

The two studies both involved small simulations (58-2048 atoms) modeled using the Lennard-Jones potential at subcritical conditions. A useful concept was used by Thompson, however, to define liquid structures during the simulation. The Cornell study defined their droplets using a computationally expensive clustering algorithm, but the Oxford study tracked the number density within the cutoff radius for each atom. Both methods yielded very similar final results. This is important because the near atoms number technique is relatively inexpensive to implement. An interesting sidelight, the study group also used a boundary condition of a closed spherical shell with externally applied repulsive walls.

Maruyama [22] essentially duplicated Thompson's work, but used the Oxford droplet definition technique exclusively and included a droplet model of water. Matsumoto et al. [23] and Ohara and Aihara [24] are two more recent similar works. All of these studies were performed within the subcritical regime for surface investigations and used relatively small models.

Koplick and Banavar [25] are physicists who just last year, 1995, published a work detailing several areas of interest in MD simulations, including droplet modeling. Again, the scales were relatively small, however, and based on subcritical models.

William G. Hoover from the Department of Applied Science at UC-Davis, has been driving engineers toward MD simulations since the early 1960's. His areas of interest lie mainly in MD simulation of gas flows, but a short description of his work [26], written in 1992, relates some interesting perspectives. His current direction is geared toward the 'combining of atomistic and continuum mechanics.' He plans to utilize MD in appropriate model areas and Lagrangian mechanics elsewhere.

As detailed in this section, a great deal of supercritical MD simulations and subcritical MD droplet studies are in progress. There are no sources, however, detailing the study of droplet evaporation in supercritical environments.

1.2.3 Parallel Methods for MD

Molecular dynamics simulations are valuable tools, but they are computationally demanding. Also, as noted, the study of supercritical diffusion requires models which are large enough to avoid an overlapping of critical point long range structures.

As a result, this study requires efficient coding on powerful computer architectures. Specifically, the capabilities of the IBM SP2 parallel computing platform were utilized.

One of the requirements for efficient parallel operations is the even loading of the computations across all the processors. Even with the relatively high densities of supercritical environments, the load imbalance on the various parallel processors due to density variations can become significant. As mentioned, the computational load is proportional to the square of the density. For a typical case, the density of the droplet is five times greater than the surrounding environment. So, for this example, the computational load in the liquid region is twenty–five times greater than in the surroundings. In response to this significant imbalance a load balanced parallel code was developed.

The algorithm described in this thesis was developed based on particle partitioning. Literature searches have uncovered several groups which developed this same concept over the past few years. Sato et al. [27] developed the concept on the Fujitsu AP1000 in 1992. They detail in their paper the load balancing advantages and the communications costs associated with the technique. They also predict the lack of scalability for large numbers of processors. Several papers by Brown et al. [28] [29] [30] detail the development of both particle and spatial decomposition algorithms. They, also, use the AP1000. Much of their work involved polymer modeling, and the inherent density variations in such structures made the particle decomposition model attractive. Kalia et al. [31], like the Brown group, developed both types of models. They preferred the spatial decomposition model, although for their small systems, both worked equally well. Their preference is based on predicted scalability.

Beazley and Lomdahl [19] discussed the utility and scalability of the spatial domain decomposition approach. They have modeled very large systems (from 1 to 600 million particles) with nearly perfect scalability on the CM-5. They do allude, however, to the problem of inefficiencies due to lack of load balancing for asymmetric problems. Also the modeling of such large systems is still a prohibitively slow process.

Plimpton [32] reviews both the particle and spatial decomposition techniques in his 1995 article. He provides an excellent description and details several other

research efforts conducted since 1988 utilizing various forms of the particle decomposition approach. He also proposes a new method called "force decomposition". Combining the load balance capabilities of the particle decomposition approach and better scaling capabilities for large numbers of processors, the force decomposition technique looks promising. For the resources readily available on the SP2, however, the particle decomposition approach proved the most useful. Further comparisons of these approaches will be provided later.

As noted, a great deal of work has been performed in the areas of supercritical combustion, molecular dynamic modeling of droplets, and parallel coding of molecular dynamics. The modeling, however, of supercritical combustion utilizing molecular dynamics has, to the author's knowledge, never before been attempted. Also, the study of systems large enough to substantiate scaling of the results to physically meaningful sizes is without precedence. The present work represents a first step toward a reliable and accurate model of supercritical and near critical combustion processes. In light of the call by combustion engineers for increased understanding and modeling capabilities in this area, the research detailed here appears quite timely.

1.3 Thesis Scope

The complete modeling of the combustion of real fuels in supercritical environments is presently beyond any single work. As previously mentioned, the diffusion of the fuel into the oxidizer environment drives the rate of combustion. So, the first step toward meeting this challenge of modeling supercritical combustion consists of developing an understanding of the evaporation in the high temperature and pressure environment. To this end, the research scope in this paper is limited to investigating the evaporation process. To simplify the study, yet still provide an important base for future research, the simulation is also limited to modeling the diffusion of a liquid droplet into a variety of subcritical and supercritical environments of the same monatomic element. The element chosen for study is argon. Having a molecular weight similar to an actual fuel, oxygen, and also having been extensively studied using molecular dynamics in the past, this is a natural choice.

The dynamic process of supercritical diffusion is poorly understood. To maximize the reliability of the results generated by this work, a minimum of simplifying assumptions are included. Three–dimensional molecular dynamic codes were developed for the transient studies. Periodic boundaries are utilized but to avoid boundary induced effects, the more spherically shaped truncated octahedron boundaries are incorporated. Also, since the surface tension is an important consideration, significant care is taken to ensure a proper initial condition of an equilibrated droplet in a supercritical environment. Details of the means of achieving this are provided later.

While a great deal of effort was expended to provide an accurate simulation of the droplet diffusion, the size limitations due to the large computational load allowed only the simulation of micro–scale droplets. The diffusion process is typically assumed as a surface phenomena. Since the surface energy of a droplet has been found to exist within only a few atomic diameters, the driving physics should still be present even in micro–scale investigations. To substantiate the validity of the work for large systems, a scaling study was also performed.

Before proceeding with the detailing of the work performed to meet the scope just defined, a review of molecular dynamics is provided. The following chapter covers fundamental concepts which were utilized to develop the simulation codes. Emphasis is placed on the description of techniques required to efficiently model large spherically symmetric systems.

More detailed molecular dynamic coding requirements are presented in Chapter 3. The implementation of parallel molecular dynamic algorithms is discussed. Also, details of an efficient coding of a spherically shaped boundary and a description of the informative code output are both provided. A unique method of dynamically tracking droplet surface energies is introduced here.

Chapter 4 provides a review of the unique aspects involved in the simulation of the supercritical diffusion process. Included is the previously mentioned explanation of the initializing process. This involves the independent modeling of droplets and environments to equilibration and then carefully fusing them together to retain the

surface energy. The chapter concludes with a description of a thermal control implemented during the long diffusion simulation. This is required to counter the latent heat effect of the evaporation process.

The results chapter begins with a review of code performance statistics. Compared to some of the best molecular dynamics codes on a variety of supercomputing architectures, the code developed for this supercritical study proved quite capable. This chapter continues with an analysis of both subcritical and supercritical diffusion simulations. The results show anticipated processes for the subcritical cases and very insightful patterns for supercritical diffusion. Promising results from the scaling research are included in this chapter as well.

As mentioned, this study is a first step toward an overlying goal of analyzing supercritical combustion. The final chapter, therefore, outlines both the work detailed herein and some related on-going efforts. The anticipated future direction of the research is also discussed.

Chapter 2

MOLECULAR DYNAMICS

The research supported by the codes presented here involves the molecular dynamic modeling of liquid droplets evaporating in gas environments. Reed and Flurchick [33] highlight the inefficiencies associated with using continuous potential (also termed soft sphere) molecular dynamics to model the gas phase. The problems stem from the low incidence of collisions. In environments near and above the critical point, however, the gas phase is very dense and the collision frequencies are much greater than in typical gas evaluations. For this reason, the codes were developed using soft spheres exclusively. This section of the report will correspondingly review continuous potential molecular dynamics.

2.1 Displacement Equations

The displacement of each atom in a molecular dynamic simulation is simply based on Newtonian particle dynamics. The atomic accelerations are a direct function of inter-atomic forces. These forces are conservative (non-dissipative) and are therefore computed as the negative gradient of the potential energy function between two atoms.

$$\mathbf{F}_{ij} = -\nabla u_{ij} = -\frac{\partial u_{ij}}{\partial r_{ij}}\,\hat{\mathbf{r}}_{ij} \tag{2.1}$$

In other words, any removal of potential energy, $-du$, is transferred without dissipative losses to work performed on the atoms, $F \cdot dr$. So knowledge of the potential between two atoms allows the analytical measure of atomic displacements.

The dashed line plot in figure 2.1 is the potential function as recommended by Maitland and Smith [34] for two argon atoms; no other atoms are present. Of course an atom within a liquid has numerous neighboring atoms influencing the potential

field at any given instant; the interactions are not simply limited to colliding pairs. Under these conditions the pairwise potential would seem to be inadequate.

The solution stems from a common simplifying assumption for soft sphere interactions: the potentials are considered 'pairwise additive' [35]. This means the force on an atom, due to interactions with surrounding neighbors, can be computed by summing the interactions with each neighbor alone. In other words, the force between two particles is considered independent of any other particles present. In essence, the problem is solved by merely assuming it away. In practice, however, a more scientific approach is applied.

To enhance simulation accuracy the potentials chosen for liquids are modified. The pairwise assumption is applied in the codes presented in this work by using an 'effective' Lennard-Jones 12-6 potential. The solid line in figure 2.1 is the effective potential for argon recommended by Allen and Tildesley [36]. As shown, this effective potential is adjusted slightly from the best estimate for pure pairwise interactions. This shifting is based on empirical liquid studies and is an attempt to account for multiple neighbor effects on the potential.

This potential, first introduced by J.E. Lennard-Jones in 1924 [35], is analytically represented by

$$u_{ij}{}^{LJ}(r_{ij}) = 4\varepsilon[(\sigma/r_{ij})^{12} - (\sigma/r_{ij})^{6}] \tag{2.2}$$

where

$$r_{ij} = |\mathbf{r}_i - \mathbf{r}_j| \tag{2.3}$$

is the separation distance between atoms i and j. The potential equation contains both short-range repulsion and long-range attraction elements. The exponent of six is based on dispersion theory and the twelve is simply chosen to match observed behavior. The value of σ is the separation distance at zero potential (approximately equal to the atomic diameter) and ε is the depth of the well on the plot. The values of $\varepsilon/k_b = 120K$ and $\sigma = 0.34nm$, as just indicated, are based on reaching agreements with experimental liquid argon studies.

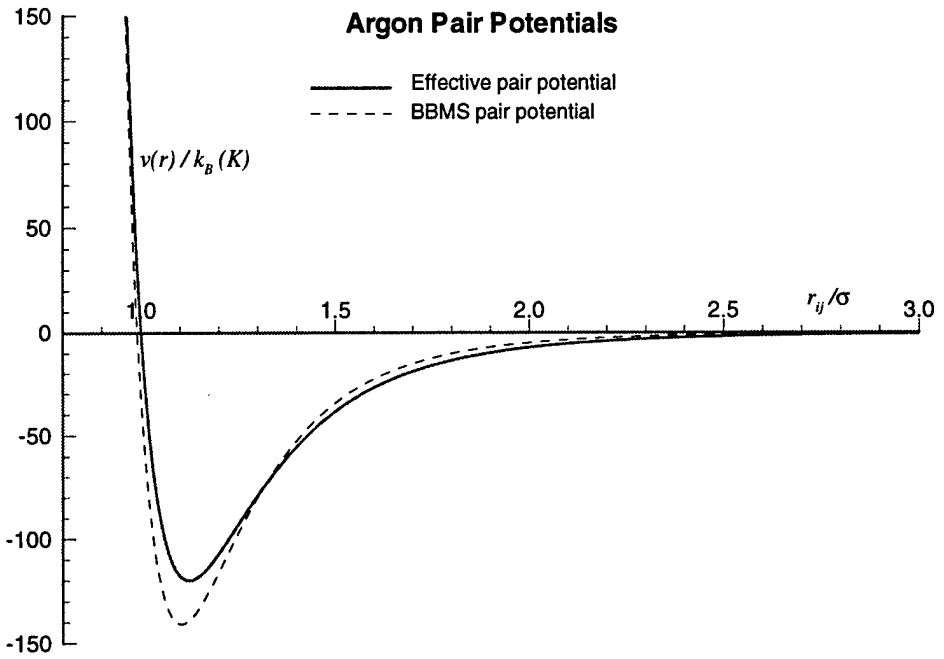


Figure 2.1. Argon pair potentials.

As mentioned, the force on one atom due to another is the gradient of the potential function. Applying this to the Lennard-Jones potential (equation 2.2) yields

$$\mathbf{F}_{ij} = \frac{24\varepsilon}{r_{ij}}[2(\sigma/r_{ij})^{12} - (\sigma/r_{ij})^{6}]\,\hat{\mathbf{r}}_{ij} \tag{2.4}$$

and noting that

$$\hat{\mathbf{r}}_{ij} = \frac{\mathbf{r}_{ij}}{r_{ij}} \tag{2.5}$$

allows the forces to be computed as

$$\mathbf{F}_{ij} = \frac{24\varepsilon}{r_{ij}^2}[2(\sigma/r_{ij})^{12} - (\sigma/r_{ij})^{6}]\,\mathbf{r}_{ij} \tag{2.6}$$

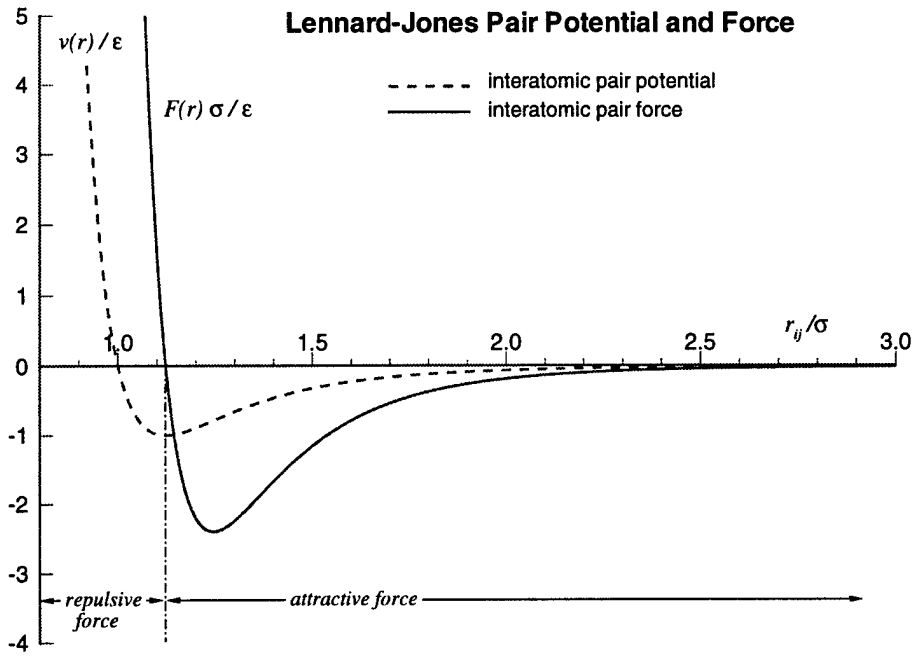


Figure 2.2. Lennard-Jones potential and force.

Pairwise additivity is assumed, and the resultant force on atom i is simply the sum of all the pairwise forces due to the surrounding j atoms.

$$\mathbf{F}_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} \mathbf{F}_{ij} \tag{2.7}$$

The relationship of the force to the potential is shown in figure 2.2 from Haile [35]. The reader should note that at a separation distance of $\sqrt[6]{2}\,\sigma$ the inter-atomic force shifts from a relatively weak attraction to a very strong repulsion. This is the mathematical representation of the energy well where the atoms in a solid lattice will reside. Many of the atoms in a liquid structure will also be found at this inter-atomic distance.

Focusing back on the displacement formulations, the reader is reminded that Newtonian particle dynamics is used to determine the atomic motions. The force on

each atom is found from equations 2.6 and 2.7. Utilizing Newton's Law, $\mathbf{F} = m\mathbf{a}$, allows the computation of the acceleration of each particle directly from the force components. Assuming this acceleration is constant across a given time increment (on the order of 10^{-15} seconds), simple explicit finite differencing yields

$$v_{i_x}^{n+1} = v_{i_x}^{n} + \Delta t \cdot a_{i_x}^{n} \tag{2.8}$$

$$r_{i_x}^{n+1} = r_{i_x}^{n} + \Delta t \cdot v_{i_x}^{n} + \frac{\Delta t^2}{2} \cdot a_{i_x}^{n} \tag{2.9}$$

where r_{ix}, v_{ix} and a_{ix} represent the x-components of position, velocity and acceleration for each displacing atom. Identical equations would result for the y and z components. These can be used to update each atom's velocity and position respectively over the time increment, Δt. Once the displacement is computed, the entire process of computing inter-atomic forces and the particle displacements can be repeated for the next time step. So a deterministic model of molecular motions can be developed using these fairly straight-forward equations.

The last paragraph is actually an oversimplification. Using such a simple finite differencing of Newton's law will often result in a non-stable molecular dynamics code. Allen and Tildesley [36] discuss the relative merits of three algorithms all designed for stability. The one selected for this work is a modification of the *velocity Verlet* algorithm, the one most highly recommended by Allen and Tildesley.

The velocity Verlet detailed by Allen and Tildesley is

$$r_{i_x}^{n+1} = r_{i_x}^{n} + \Delta t \cdot v_{i_x}^{n} + \frac{\Delta t^2}{2} \cdot a_{i_x}^{n} \tag{2.10}$$

$$v_{i_x}^{n+\frac{1}{2}} = v_{i_x}^{n} + \frac{\Delta t}{2} \cdot a_{i_x}^{n}$$

$$a_{i_x}^{n+1} = f(r_{ij}^{n+1})$$

$$v_{i_x}^{n+1} = v_{i_x}^{n+\frac{1}{2}} + \frac{\Delta t}{2} \cdot a_{i_x}^{n+1}$$

where $f(r_{ij}^{n+1})$ refers to the force computation function of equation 2.7. This finite difference form adds stability by making the velocity computation semi-implicit. Half

of the velocity update is computed using the acceleration at the previous time step. This is an explicit formulation. The solution of the velocity is completed, however, using the updated acceleration value. This step-wise solution of the velocity makes the algorithm very stable, but still quite simple.

The modification used in the present work is simply a reordering of the steps to remove the necessity to save acceleration information between time steps. Assuming the initial position is at time step $n+1$ and the initial velocity is at $n+\frac{1}{2}$, the formula details as follows

$$a_{i_x}^{n+1} = f(r_{ij}^{n+1}) \tag{2.11}$$

$$v_{i_x}^{n+1} = {v_{ix}}^{n+\frac{1}{2}} + \frac{\Delta t}{2} \cdot a_{i_x}^{n+1}$$

$$r_{i_x}^{n+2} = r_{i_x}^{n+1} + \Delta t \cdot v_{i_x}^{n+1} + \frac{\Delta t^2}{2} \cdot a_{i_x}^{n+1}$$

$${v_{ix}}^{n+\frac{3}{2}} = v_{i_x}^{n+1} + \frac{\Delta t}{2} \cdot a_{i_x}^{n+1}$$

This does not change the accuracy or stability of the algorithm; the computations are identical. The change simply reduces the memory requirements and array access workloads since the acceleration is discarded between time steps.

2.2 Numerical Efficiency

Regardless of the specific displacement algorithm chosen, the bulk of the model computations will be found in the force evaluations. At the elementary level, each atom will see a pairwise contribution from every additional atom in the system. This equates to an order of N^2 computations ($O(N^2)$), where N is the number of particles in the system. For models of more than a few thousand atoms, this level quickly exceeds the capabilities of the largest supercomputers. Fortunately, there are several simplifying assumptions and techniques available to reduce the $O(N^2)$ workload.

Close inspection of figure 2.2 reveals that the attractive force reduces asymptotically to zero. For argon, the force becomes negligible at distances of just 2.5σ. At a liquid density, the number of neighbors which fall within this cutoff radius is

on the order of fifty. So, even for a dense system, the force computation reduces to $O(50N)$. This alone appears to solve our problem, but there is still a factor driving the force computations to $O(N^2)$.

The efficiency of the force cutoff just mentioned can only be applied if the neighboring atoms are identified. The computations required for this are

$$r_{ij_x} = r_{i_x} - r_{j_x} \tag{2.12}$$

$$r_{ij_y} = r_{i_y} - r_{j_y} \tag{2.13}$$

$$r_{ij_z} = r_{i_z} - r_{j_z} \tag{2.14}$$

$$r_{ij}^2 = r_{ij_x}^2 + r_{ij_y}^2 + r_{ij_z}^2 \tag{2.15}$$

where r_{ij}^2 can be compared to the square of the cutoff radius. While there are only eight operations here, they still must be applied for all pairs of atoms. Therefore this $O(N^2)$ distance check will dominate the computation load.

Verlet proposed in 1967 a neighbor list approach [37] where the $O(N^2)$ search is performed to generate an array set containing neighbor identifications. This set consists of a pointer array, with an element for every atom, and a list containing the complete set of neighbors for every atom. The pointer values direct the search to the starting locations in the neighbor list of a set of atoms within an enlarged cutoff radius, r_{list} (see figure 2.3). This list array requires significant memory resources, but it remains useful until an atom traverses the gap between the cutoff and the list radii. During this period the $O(N^2)$ computations are avoided by using the list as an alternate source for neighbor candidates.

A large list sphere appears desirable; the number of time steps elapsed before an atom passes through the list gap is greater. But every step still requires a cutoff check. If the list radius is too large, the sampling cost becomes prohibitive. For simulations of monatomic liquids, a value of 2.8σ yields the best performance with an update frequency of around twenty steps. In short, while the technique reduces the workload, the $O(N^2)$ problem remains, and the size of the neighbor list array can become problematic regarding computer memory requirements.

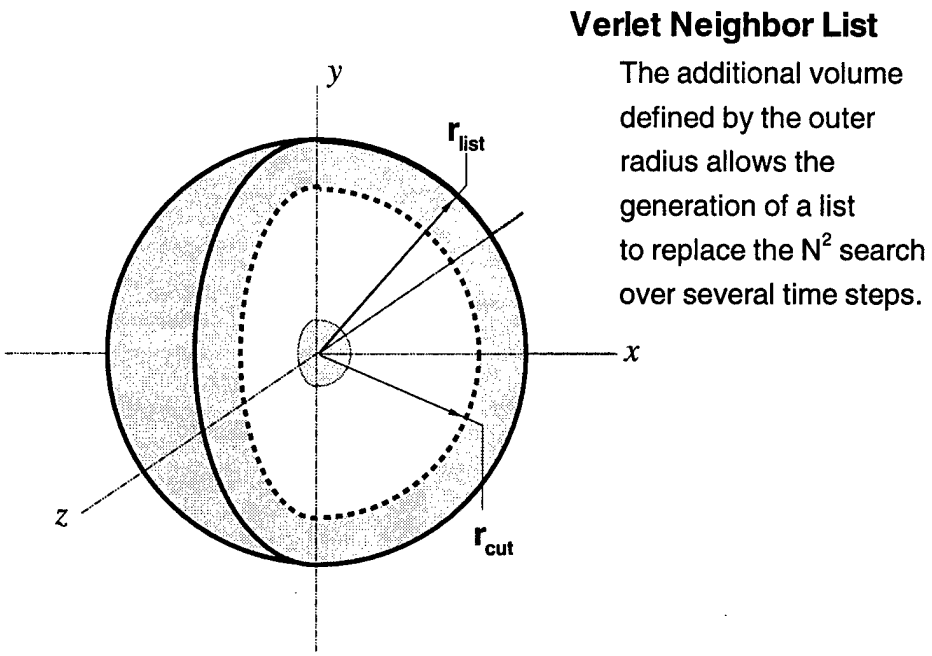


Figure 2.3. Verlet neighbor list.

In 1975, Quentrec and Brot [38] reduced the problem to O(N) by griding the spatial domain into cells whose sides are equal in length to the cutoff radius. The particles are then tagged in a set of cell arrays based on their computed locations. Figure 2.4 shows the grid concept. The generation of the cell arrays, `head()` and `llist()`, for an $m \times m \times m$ grid is performed as follows

```
      msq = m*m
      mcube = m*msq
      rm = real(m)

      do 10 icell=1,mcube
        head(icell) = 0
   10 continue

      do 20 i=1,N
        icell = 1 + int(rx(i)*rm/boxlength + rm/2)
                  + int(ry(i)*rm/boxlength + rm/2) * m
                  + int(rz(i)*rm/boxlength + rm/2) * msq
```

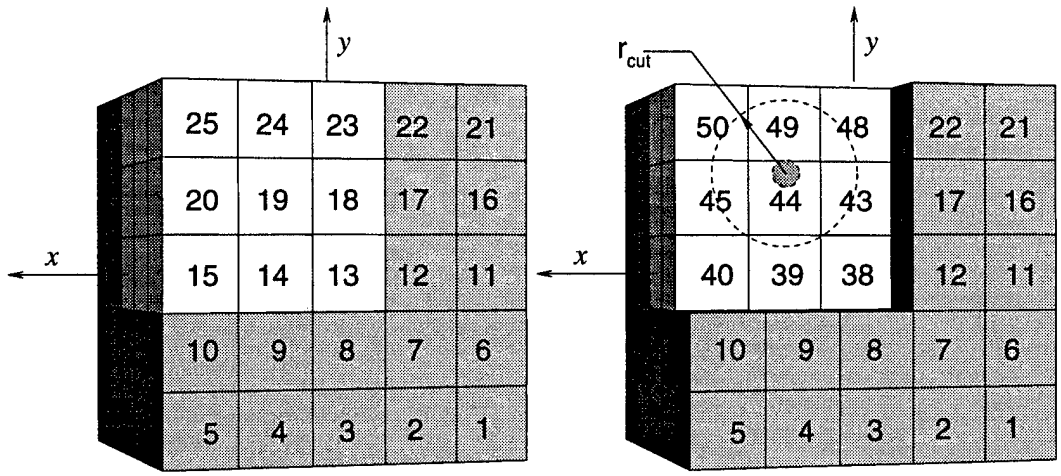


Figure 2.4. Linked list grid structure. The grids are numbered in ascending order along x, y and z coordinates respectively. The cutoff sphere for the atom in cell 44 is fully covered using the highlighted 27 cell substructure.

```
      llist(i) = head(icell)
      head(icell) = i
20 continue
```

The boxlength is the length of a side of the computational domain and the `int()` function uses the conventional rules of Fortran 77 integer truncation. This is an O(N) task, and, once performed, limits the search pattern to a particle's cell and the twenty-six surrounding cells.

This technique is designated 'linked list' by Allen and Tildesley [36]. The terminology refers to the linking of the list array with the head array during the neighbor search. The head array, with an element for every cell, contains the largest atom number within each cell (or zero if there are no atoms). The list has N elements and acts as a self-pointer to the next lower atom number. If there are no atoms left in the cell, the list value will be zero. This search technique is illustrated in figure 2.5. As evidenced by the relatively small array pair, an additional benefit associated with this approach is the efficient use of memory to track the neighbor locations.

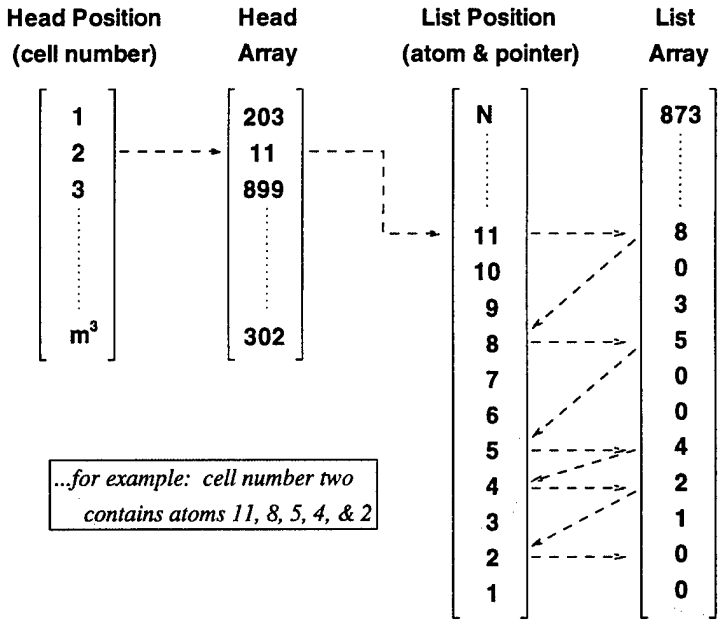


Figure 2.5. Linked list array set. The header gives the first atom number in the cell and points to the location in the list of the next atom number. The list then acts as a self-pointer for the remaining atoms in the cell.

While the linked list is a significant enhancement over the Verlet neighbor list, it does have a drawback. The size of the inspection volume for cutoff checks is

$$V_{\substack{linked \\ list}} = (3r_{cut})^3 = 27r_{cut}^3 \tag{2.16}$$

The cutoff volume, however, is only

$$V_{cutoff} = \frac{4}{3}\pi r_{cut}^3 \approx 4.2r_{cut}^3 \tag{2.17}$$

So the inspection volume is six times larger than necessary. By contrast, the inspection volume for the optimal monatomic Verlet list is $5.9r_{cut}^3$, only 40% greater than the cutoff volume. As mentioned, however, the neighbor list concept requires an

$O(N^2)$ update. For large systems, the linked list approach is therefore superior, but one further refinement is still available.

The solution to the drawbacks of both the linked list and the Verlet neighbor list is interestingly found by combining them. The linked list cells can be set to the size of r_{list} ($\approx 1.12 r_{cut}$). This increases the inspection volume to $38 r_{cut}^3$, nine times larger than desired. But the generation of a Verlet list during the initial pass delays the update of the linked list for twenty steps. Also, for liquid structures, this linked list workload is $O(450N)$. So for systems of 500 atoms or greater, the inclusion of the linked list removes the $O(N^2)$ search of the neighbor list.

Since the systems modeled herein range in size from 20,000 to over 350,000 particles, this combined approach is utilized. The inclusion of the neighbor list, though, for these large cases presents a memory management challenge. The parallelization of the technique provides the solution; details of this are provided in the next chapter.

So the handling of large simulations on parallel processors appears attainable. The simulation size does, of course, have a limit, and this limit is still very small from a physical standpoint. The inclusion of simulation boundaries which allow the most efficient use of simulation volume is, therefore, very important. The following section details both general MD boundary conditions and a more unique boundary chosen for the work presented herein.

2.3 Boundary Conditions

The study of droplet diffusion does not require the evaluation of solid boundaries. In fact, to reduce the effects on the diffusion simulation, a solid wall would require a larger domain (by as much as twenty atomic diameters [35]). Instead, a common boundary condition used in molecular dynamics is the *periodic boundary.* This boundary technique is utilized throughout the work presented herein.

Figure 2.6 demonstrates how this approach models a system as an infinitely repeating physical domain. If a molecule is close to the right side of the domain, it will *see* the molecules on the left side as neighbors. Also, if a molecule leaves the physical domain, it will re-emerge on the opposite side. This approach does not, however, remove the need for systems large enough to model the long-range order of

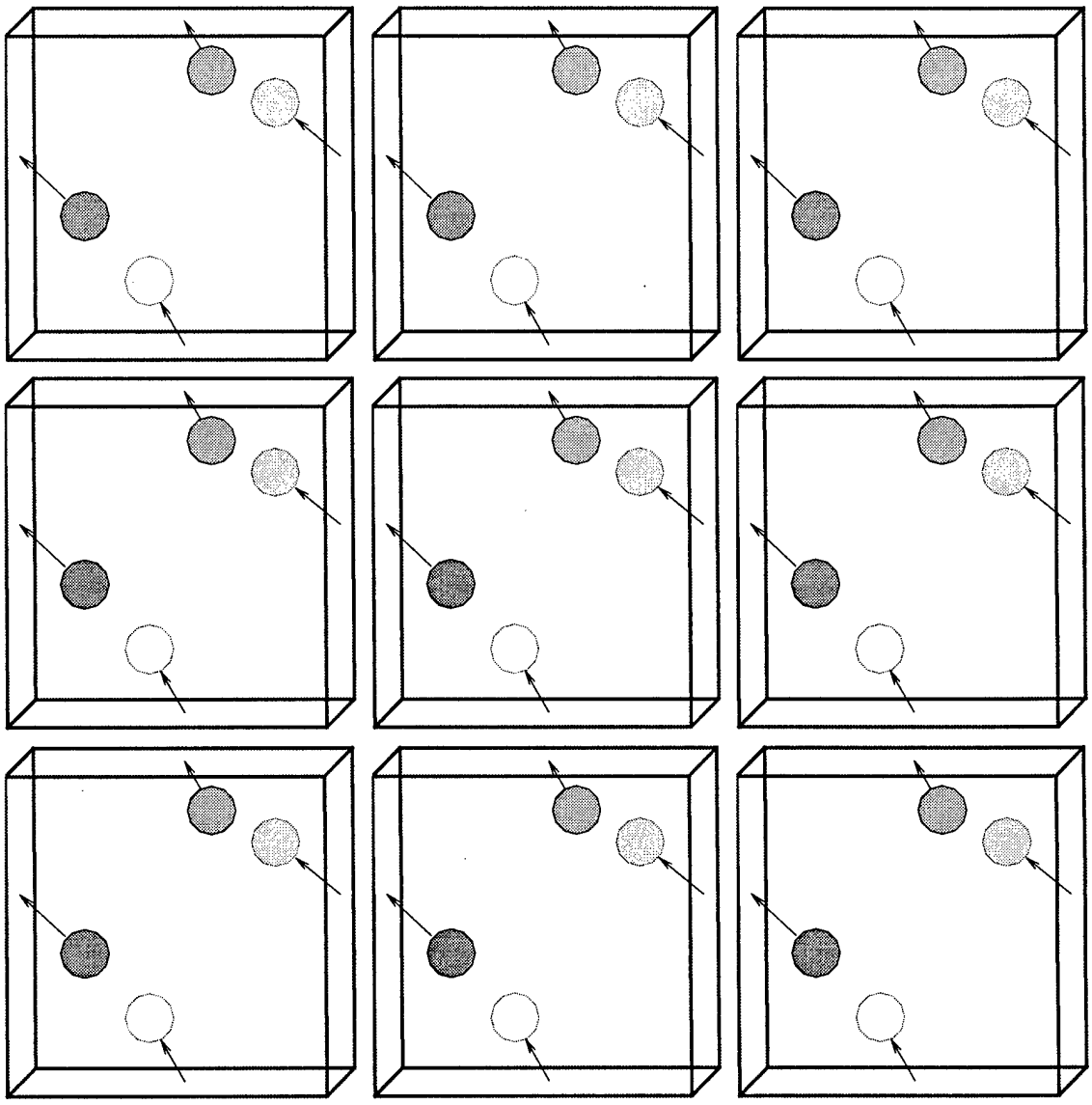


Figure 2.6. Periodic cubic boundaries.

supercritical fluids. If the long-range order wraps over upon itself the simulation will, of course, be invalid.

For the large systems required, periodic boundaries can become costly. The determination of the re-imaging of the displaced atoms is relatively cheap; this is performed outside of the force loop. We are not so fortunate, however, with respect to the inter-atomic distance corrections. The required logic is simply

$$r_{ij_x} = \begin{cases} r_{ij_x} & \text{if} \quad -\frac{1}{2}L_{box} < r_{ij_x} < +\frac{1}{2}L_{box} \\ r_{ij_x} - L_{box} & \text{if} \quad +\frac{1}{2}L_{box} < r_{ij_x} \\ r_{ij_x} + L_{box} & \text{if} \quad -\frac{1}{2}L_{box} > r_{ij_x} \end{cases} \tag{2.18}$$

$$r_{ij_y} = \begin{cases} r_{ij_y} & \text{if} \quad -\frac{1}{2}L_{box} < r_{ij_y} < +\frac{1}{2}L_{box} \\ r_{ij_y} - L_{box} & \text{if} \quad +\frac{1}{2}L_{box} < r_{ij_y} \\ r_{ij_y} + L_{box} & \text{if} \quad -\frac{1}{2}L_{box} > r_{ij_y} \end{cases} \tag{2.19}$$

$$r_{ij_z} = \begin{cases} r_{ij_z} & \text{if} \quad -\frac{1}{2}L_{box} < r_{ij_z} < +\frac{1}{2}L_{box} \\ r_{ij_z} - L_{box} & \text{if} \quad +\frac{1}{2}L_{box} < r_{ij_z} \\ r_{ij_z} + L_{box} & \text{if} \quad -\frac{1}{2}L_{box} > r_{ij_z} \end{cases} \tag{2.20}$$

The coding is straight forward, but this check must be applied to all pairs prior to computing the force interactions. It is part of the step where the potential cutoff determination is made. As noted in the previous section, just a few operations during this highly repetitive section are very costly. The logic in equations 2.18 to 2.20 is equivalent to about thirty operations on the SP2 for every inter-atomic distance check. So removing this load from the simulation, or at least reducing its impact, is highly desirable.

A partial solution to this problem has been developed and incorporated into the codes presented herein. Each code includes a cell map which allows the logic to be *cell aware.* All cells on the boundary include the costly logic, but elsewhere the steps are left out. Since the cells are at least the dimension of the potential cutoff radius, interior cell atoms will not experience a cross boundary interaction. In effect, the

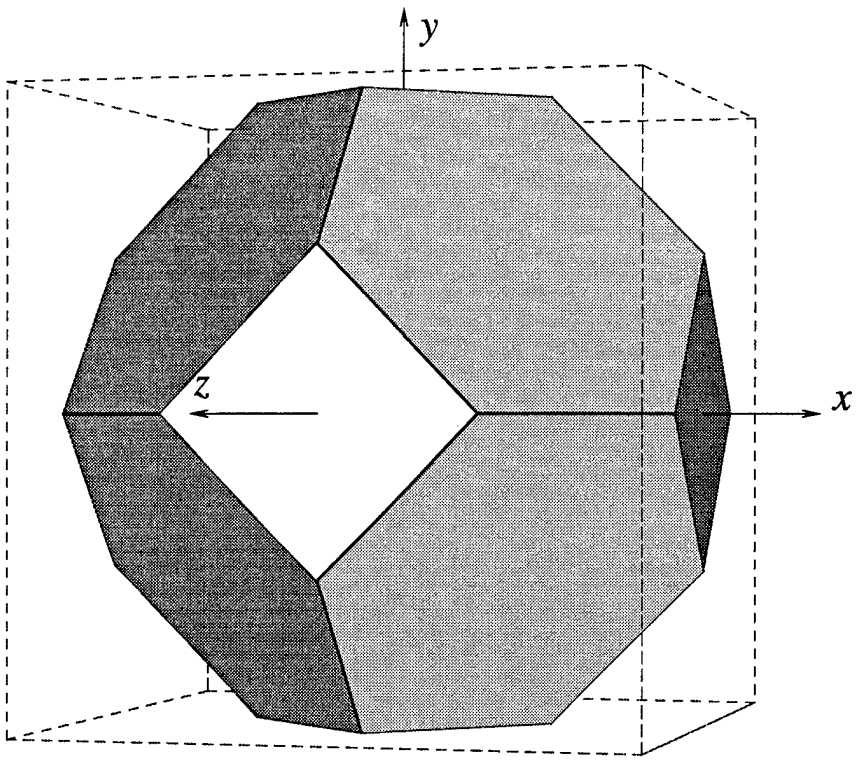


Figure 2.7. Truncated octahedron computational domain.

additional logic normally required in the interior cells is determined and included in advance in the cell map.

Before leaving this section, there is one more concern to cover. The periodicity demonstrated in figure 2.6 is for the most common boundary used in molecular dynamics, the cube. Since the work presented herein involves a process which predominantly occurs on the droplet surface, spherical periodicity is preferred. This would allow an even distance between the droplets and therefore balance any image effects. Also, this shape would maximize the droplet image distances (and therefore minimize image effects) for a given environment volume. Unfortunately a sphere is not a periodic shape (i.e. cannot be used to tile a 3-D space). There are other boundaries which are available, however.

A boundary condition termed the *truncated octahedron* comes much closer then the cube to representing the desired shape. The truncated octahedron is shown in

figure 2.7. The included sphere in a cube accounts for only 52% of the cube's volume. So nearly half of the environment volume is wasted additional simulation. For a truncated octahedron, the included sphere requires 68% of the simulation volume. Perhaps even more important, the truncated octahedron has a much more desirable image ratio. This term refers to the ratio of the largest to the shortest distance between periodic images. The optimum value to reduce the effect of the periodicity on the simulation is one. For a cube this is the ratio of the cross-diagonal to the length of a side, 1.73. The truncated octahedron has an image ratio of just 1.29, by far the best of all the known periodic boundary candidates [39].

The periodics of the truncated octahedron shape, however, are not as easily understood as the cubic periodics. This latter form places the drops in a cubic lattice orientation in space. The truncated octahedron places them instead in a body-centered layout. Figure 2.8 shows the honeycomb three-dimensional pattern of the body-centered periodic images by displaying four images on top of a layer of nine. The shapes are still periodic on all six axis faces. This fact is more clearly seen when only one layer of the honeycomb is viewed in figure 2.9. When an atom crosses a hexagon plane, however, it must enter the next layer of the honeycomb. Figure 2.10 represents this by detailing only the movement of the atoms from the back layer (solid lines) to the next layer (dashed lines). The atom, of course, does not actually move outside the domain when it crosses the hexagon plane; instead it re-images cross-diagonally. (Further details of this re-imaging are provided in the next chapter.)

This periodic boundary also has an even more adverse affect on the computational load. The logic to correct the inter-atomic distances is a two step process [40]. The first is identical to the cubic boundary where the interactions crossing the axis planes are checked using equations 2.18 through 2.20. The additional step involves the correction across the hexagon planes. The required logic is

$$\text{if } \{|r_{ij_x}| + |r_{ij_y}| + |r_{ij_z}|\} > \tfrac{3}{4} L_{box} \text{ then} \tag{2.21}$$

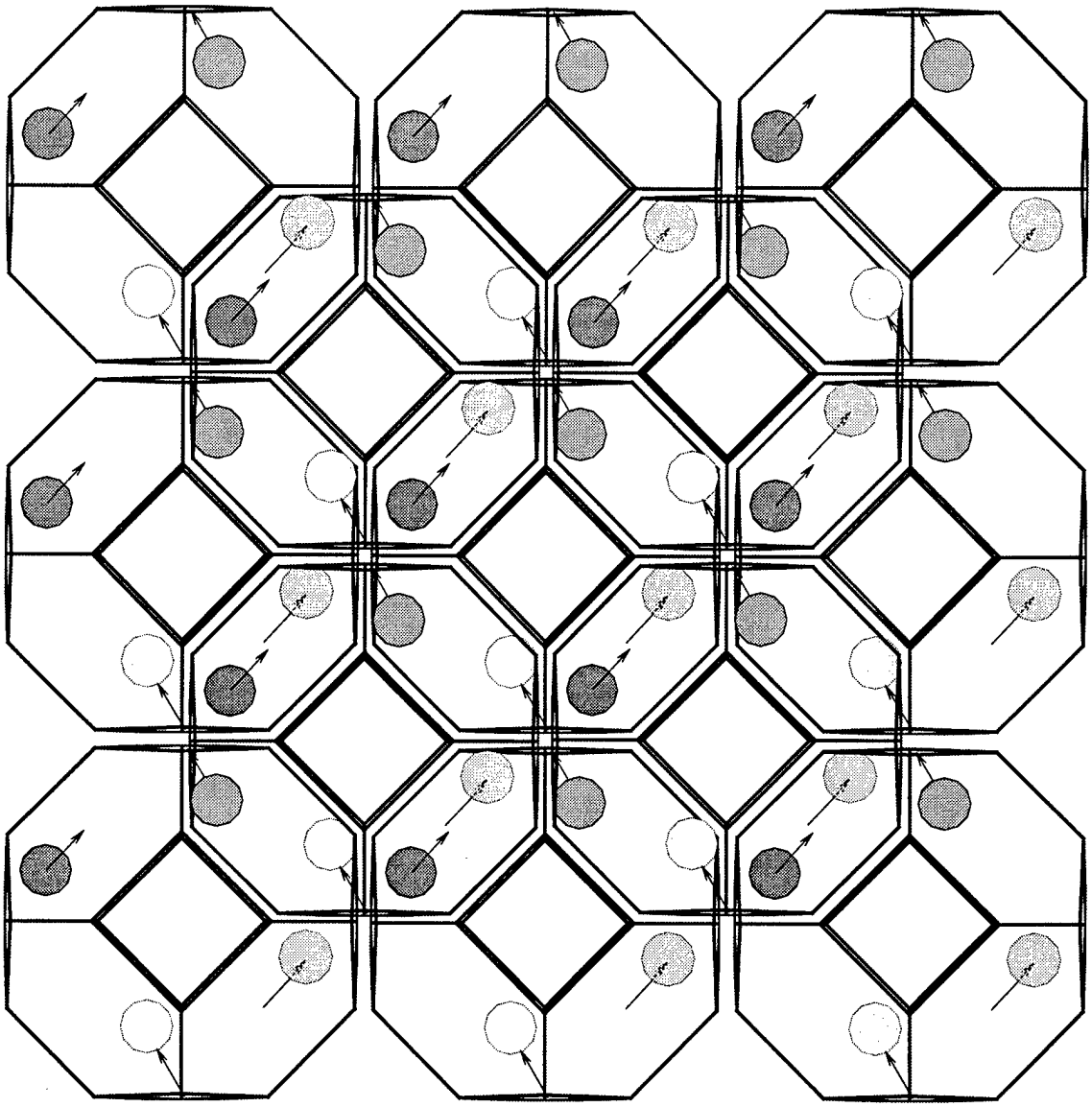


Figure 2.8. Truncated octahedron periodic structure.

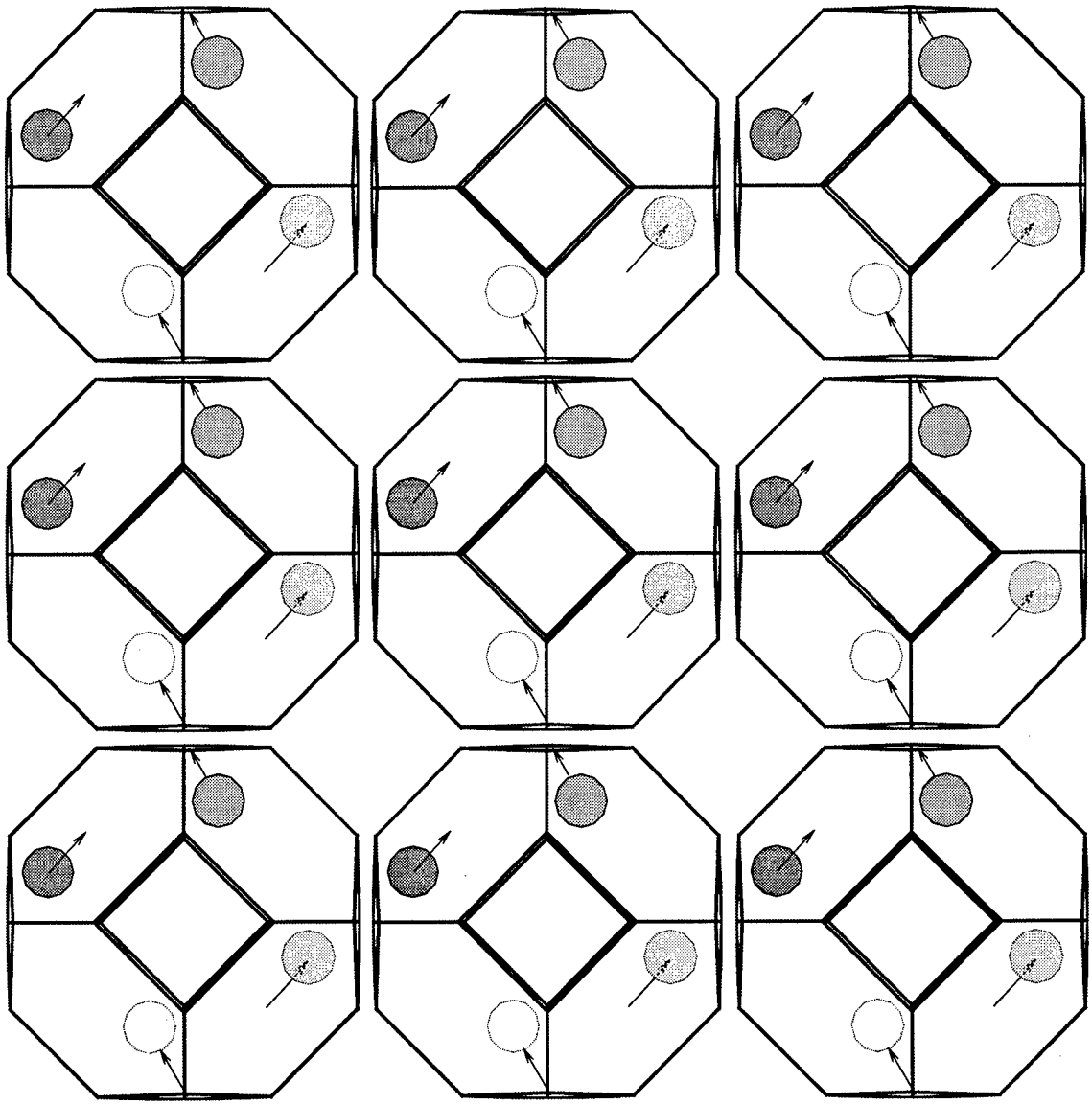


Figure 2.9. Truncated octahedron axial periodics. The particles still re-image across planes perpendicular to the x, y and z axes.

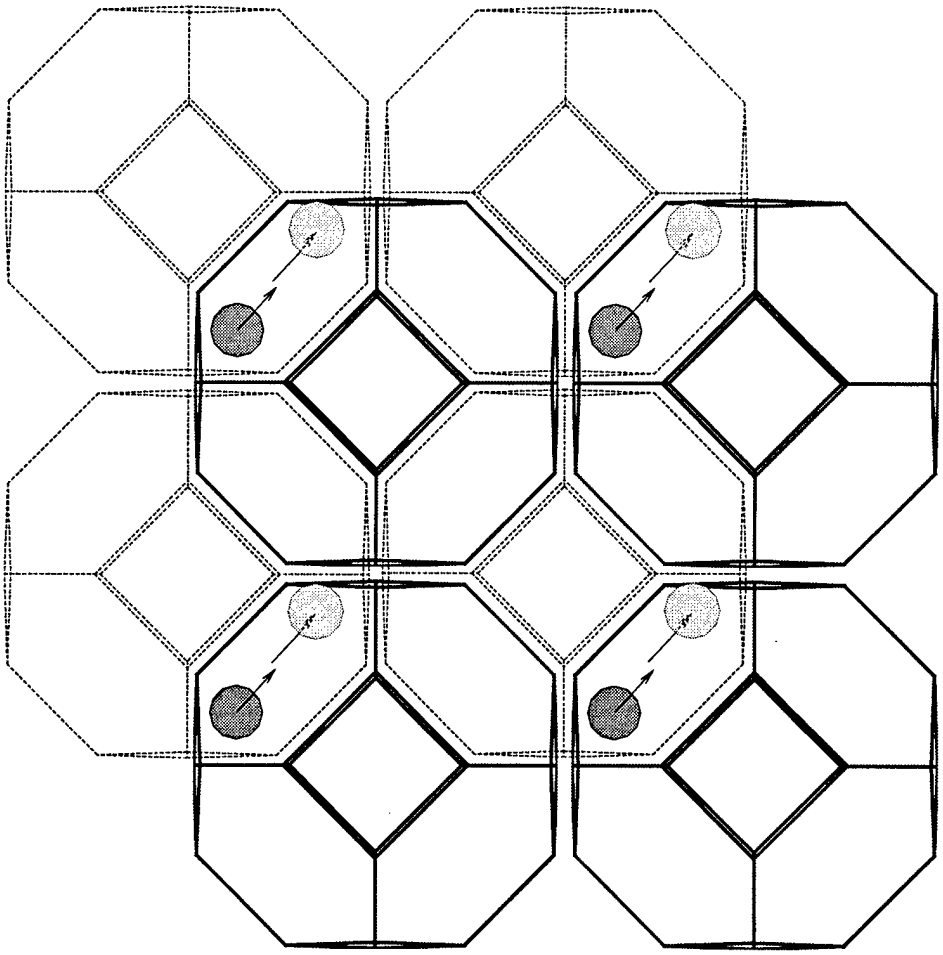


Figure 2.10. Truncated octahedron cross-quadrant periodics. Atoms which cross the hexagon planes of the periodic shape re-image in the opposite quadrant.

$$r_{ij_x} = \begin{cases} r_{ij_x} + \frac{1}{2}L_{box} & \text{if } r_{ij_x} < 0 \\ r_{ij_x} - \frac{1}{2}L_{box} & \text{if } r_{ij_x} \geq 0 \end{cases}$$

$$r_{ij_y} = \begin{cases} r_{ij_y} + \frac{1}{2}L_{box} & \text{if } r_{ij_y} < 0 \\ r_{ij_y} - \frac{1}{2}L_{box} & \text{if } r_{ij_y} \geq 0 \end{cases}$$

$$r_{ij_z} = \begin{cases} r_{ij_z} + \frac{1}{2}L_{box} & \text{if } r_{ij_z} < 0 \\ r_{ij_z} - \frac{1}{2}L_{box} & \text{if } r_{ij_z} \geq 0 \end{cases}$$

The Fortran `sign` function can be used to reduce some of this logic as follows

```
if((abs(rxij)+abs(ryij)+abs(rzij)).gt.(.75*boxlength)) then
  rxij = rxij - sign(.5*boxlength,rxij)
  ryij = ryij - sign(.5*boxlength,ryij)
  rzij = rzij - sign(.5*boxlength,rzij)
endif
```

But this is still an additional workload added to the expensive force loop.

An effective work-around of this additional computational load has been developed as part of this research. Cell aware coding is again utilized, but the solution is also based on the ability to include both the main truncated octahedron boundary and all of the cross diagonal image volumes within a single cubic shape. The linked list array can then contain image data in addition to the primary information. Details of this unique approach are reserved for the next chapter.

This has been a brief introduction to the concepts associated with molecular dynamics modeling. In short, the primary computation workload is associated with determining the force on each molecule. Periodic conditions remove wall effects, but do not remove the need for systems large enough to include any long-range orders present. Also, special techniques, such as the linked list sorting, have successfully reduced the computational requirements to O(N); thus allowing larger scale evaluations. No mention has been made of parallel molecular dynamic coding. This subject and other detailed concepts directly related to the problem at hand are pursued next.

Chapter 3

CODE DEVELOPMENT

There are numerous established codes available to perform molecular dynamic simulations. Most of these are very complex programs utilized mainly by chemists to investigate molecular structures and interactions of complex molecules. *Amber* [41], for example, is actually a series of programs which includes a detailed force field library. Containing recently updated fields for proteins and nucleic acids, the code is intended for biomolecular dynamics. Such complicated force fields are unnecessary for the present work; the potential functions for argon are well established and straightforward. There are challenges, however, associated with this research for which the solutions were not well versed. Properly handling the spherical geometry of the initial droplet condition, dynamically visualizing the shape and temperature profile, and determining the effect of the supercritical environments on the droplet surface tension were all challenges of this nature. In pursuit of their solutions, an in-house code development ensued.

This chapter details the unique approaches which were included in the resultant code. From the start, parallel strategies were utilized to provide enhanced speed for large systems. A discussion of a variety of parallel techniques is therefore presented. The previous chapter discussed the solution of using truncated octahedron boundaries to properly model the spherical nature of the diffusion. The discussion is continued here with a description of the means of efficiently implementing such boundaries in the simulation code. This chapter concludes with a detailing of property computations. A variety of information is generated. Among the highlights is the ability to track the interior droplet structure during the simulation. Also a dynamic measuring and visualizing of the density, the temperature and the surface tension provided important insight concerning the diffusion process. The measuring of the surface tension is, as far as the author is aware, a unique method.

3.1 Parallel Aspects

Molecular dynamics simulations involve very large computation loads. Every atom in a liquid simulation interacts with an average of fifty neighboring atoms at each time step. Also, the accurate modeling of the collision dynamics is an essential ingredient required for simulation accuracy. Since these collisions occur over very small time frames, the temporal discretization requires step sizes on the order of only a few femtoseconds (10^{-15} seconds). The large loads, both incremental and overall, would seem to make molecular dynamics a very good candidate for parallel computations. The unstructured nature of MD, however, significantly complicates the achievement of efficient performance.

Many traditional computational models of fluid flow are established from the Eulerian view. In other words, the flow is investigated from a series of fixed computational windows as it moves through the simulation. This allows the setting of a stationary series of discretized points in space for the evaluation. If there is a high density point of interest, the discretization is compressed here, but the points are still predetermined. When such evaluations are structured for parallel coding the natural decomposition of the workload is spatially based. This allows an even loading of the computations with the spatial zones assigned equal numbers of grid points. An added benefit from this division also results. The communications between processors are proportional to simulation areas. This will be shown shortly to be very beneficial.

One approach to parallel molecular dynamics is based on the same spatial decomposition, or at least almost the same. Fixed simulation volumes are selected as computational boundaries. Denser regions, as before, can be split into more numerous zones. But now each atom represents a modeling point, and these points are floating through the simulation. Not only are the MD grid points, the atoms, communicated as neighbors across the computational boundary, they also can actually move into a new processing volume. This complicates the coding but does not eliminate this decomposition approach as a viable parallel tool. The communications are still area based and a study by Plimpton [32] shows this approach to have the greatest potential efficiency for MD modeling. There are two other approaches, however, that have more desirable characteristics for the problem detailed herein.

The other decomposition strategies are termed "atom decomposition" and "force decomposition" by Plimpton. Within this paper, atom decomposition will also be called particle decomposition interchangeably. As just mentioned, the spatial decomposition approach offers the most promise for efficient parallel computations. The code developed within this paper, however, does not use spatial decomposition since it would have poor load balancing.

The computational load for molecular dynamics has already been established in the previous chapters as proportional to the square of the atomic density. So, unless the system is uniformally dense, processors cannot simply share the same size geometries to evenly share the load. Setting variable geometries to ensure the same number of atoms in each space is not enough; the average density in each region must be the same. Even if an initial balance is established, just a small shift in density profiles can be seriously detrimental. Remember the load is not simply density based, it is proportional to the square of the density. Since the problem at hand consists of a dynamically changing density profile, alternatives to the spatial decomposition approach were investigated.

Particle decomposition was the technique ultimately used. Although originally envisioned as a unique approach, the first chapter noted that many other researchers have independently begun using essentially identical decompositions. Perhaps the reasons for this spontaneous development are twofold: load balancing is achieved and the technique is easy. The following paragraphs review the approach.

In an attempt to develop an evenly loaded code, the balancing of the average density for each processor for the life of the simulation was pursued. As a first step, each processor was assigned an equal number of atoms for displacement computations. As mentioned, this alone is not sufficient. If, however, the initial assignment is performed with a uniformity of atoms across all density profiles, the configuration ensures balance. This concept is depicted in figure 3.1. A central slice of data from an actual simulation run is presented. As shown, the average density profiles across the processors are nearly identical. The similarity ensures an even computational balance since the squares of each partition's density are essentially equal.

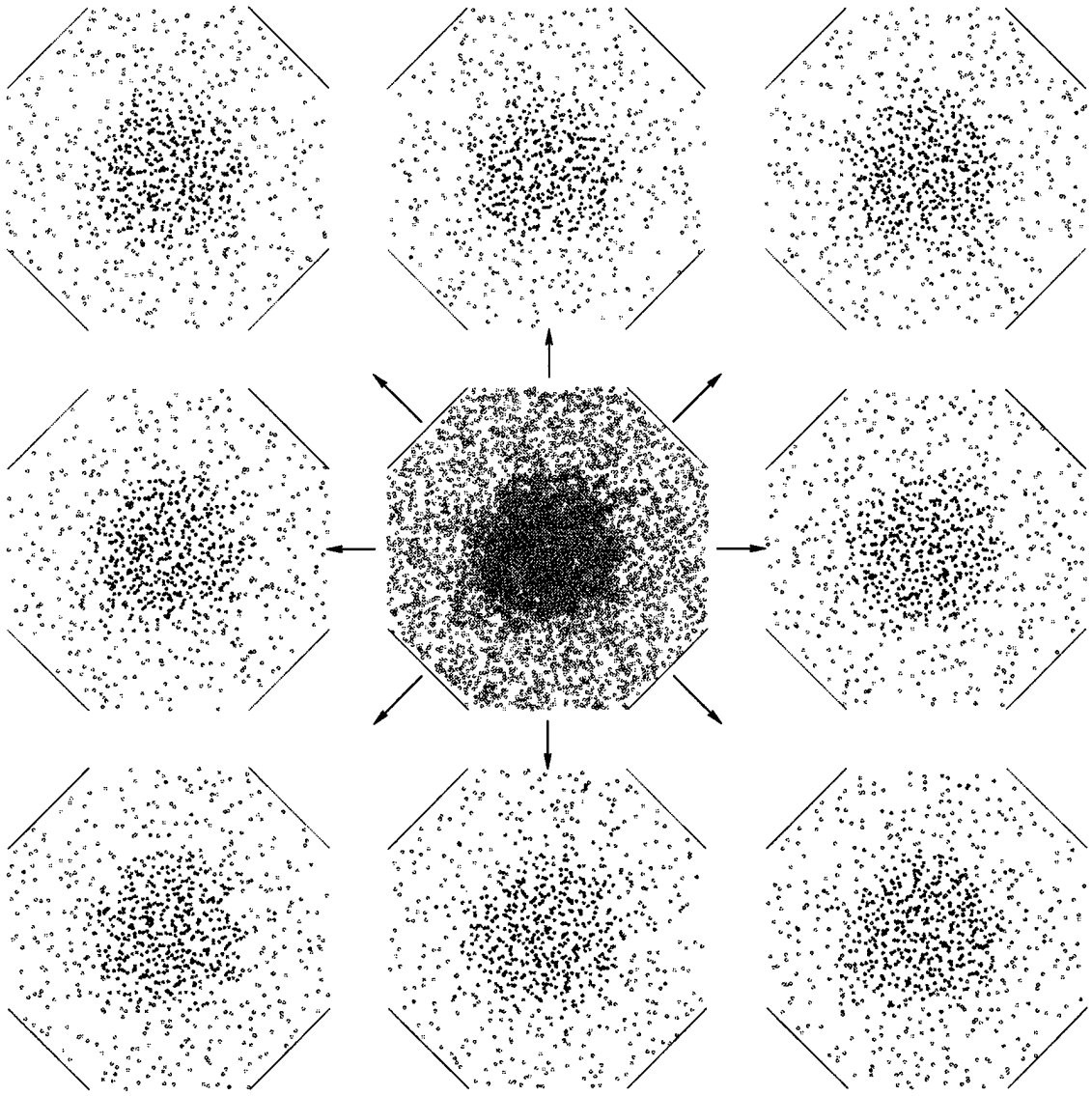


Figure 3.1. Atom decomposition. The central image is a slice of data from a simulation of a 27,000 atom drop in a 64,000 atom environment. The simulation is modeled across eight processors and evenly balanced as shown.

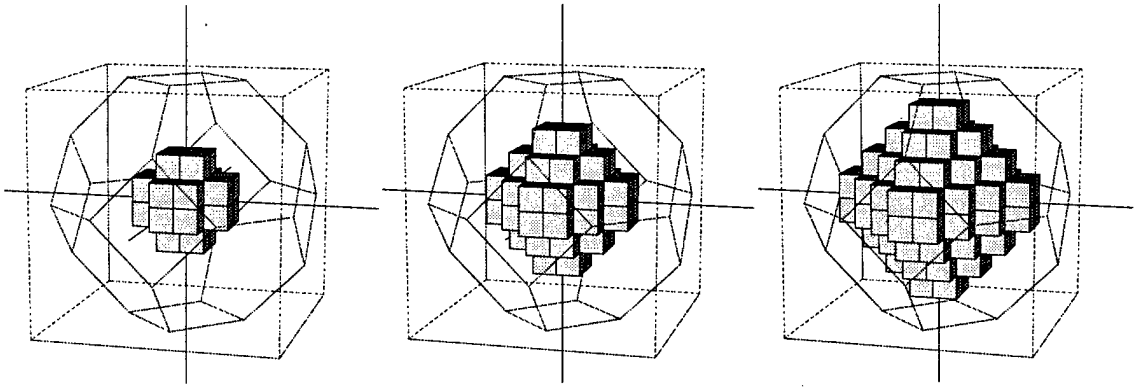


Figure 3.2. Cell map order. The progression from the dense center outward allows the setting of the initial load balance.

The initial balance could be set using a random selection, but the cell map structure mentioned in the previous chapter provides a more certain method. Figure 3.2 shows the order in which atom displacements are computed for the truncated octahedron shape. The linked list cells are stepped through an expanding staircase structure from the dense interior of the simulation outward to the low density surroundings. Using this same order, the initial atoms can be assigned evenly across the processors by a simple distribution. An additional benefit is the ability to reload a simulation onto a different processor set level if desired. A run begun on eight nodes, for example, could be restarted on thirty-two and still be perfectly balanced across the expanded set.

This balance could also be initially set for spatial decomposition algorithms as well by using variable sized geometries. The strength of the new approach, though, is that as the simulation progresses, the average density profile is maintained by the chaotic nature of the atomic motions. If statistically high enough numbers of atoms are assigned to each processor, the profiles will remain similar for each partition. In other words, there is no driving atomic force which would cause one processor to see, on average, any different profile than the others.

The particle decomposition technique therefore solves the problem of load balancing with apparent ease. There is still a challenge, however, associated with this approach. If the atoms assigned to each processor consist of a set that are found uniformally across the entire physical domain of the simulation, then the atomic

neighbors also reside throughout the system. The location of these neighbors are required for the simulation. Determining upon which processor each neighbor resides is seemingly an insurmountable problem. There is, however, a simple solution for this as well.

If each processor contains the complete global set of atomic positions then the problem disappears. The determination of which processor displaces a neighbor is irrelevant; only the location of the neighbors is required. After each time step, the algorithm therefore communicates atomic positions across all the processors. This sets the global position array and readies the simulation for the next series of displacement computations. The communication loads associated with this step, however, generate a cost which is not so easily avoided.

The logic required to perform the communication is actually quite simple. Global message passing tools are available to achieve in just a few lines of code the required process. The Message Passing Library (MPL) found on the IBM SP2 provides the following *concat* command [1]

```
call mp_concat(rx,grx,npr8,allgrp)
call mp_concat(ry,gry,npr8,allgrp)
call mp_concat(rz,grz,npr8,allgrp)
```

These three simple lines of code transfer the local position arrays, `rx`, `ry` and `rz`, into global arrays, `grz`, `gry` and `grz`, which are then communicated across all processors. The `npr8` variable communicates to the system the size of the arrays and `allgrp` is a processor group designation. This is graphically represented in figure 3.3.

This division of local and global arrays is also easily incorporated into the linked lists and Verlet neighbor lists. For the linked list, local head and list arrays are generated using the r_x, r_y and r_z local arrays while global linked list arrays are generated using gr_x, gr_y and gr_z. The displacing atom coordinates are found using the local lists while the neighbor candidates are determined using the global linked list set. The Verlet list is even easier. In fact, the inherent memory saturation problem of this list is actually resolved by the parallel technique. The neighbor list is generated

[1] An equivalent call in the parallel standard *Message-Passing Interface* (MPI) library is the *all-gather* operation [42].

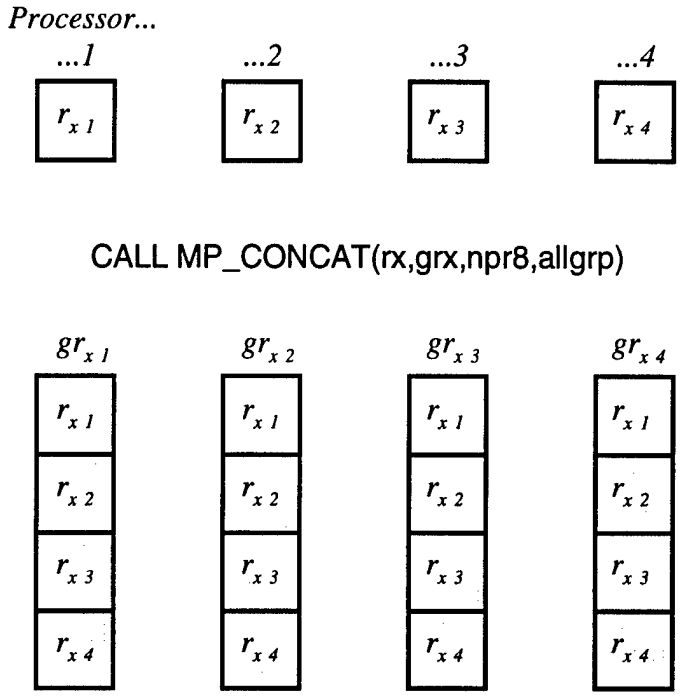


Figure 3.3. The *concat* command. A collective communication tool in the *Message Passing Library* on the IBM SP2.

identically as before but is automatically divided evenly among all of the processors since the pointer elements are likewise separated.

The cost, then, is not complicated coding. In fact, this technique can be implemented on existing serial MD codes quite easily. The cost refers to the fact that the communications at every time-step involve every atom in the simulation. This means the amount of communications is directly related to N, the total number of atoms, and therefore proportional to the simulation volume. If a problem is divided among additional processors, the simulation volume, and therefore the communication loads as well, remain constant. The computations are reduced, but eventually the communication costs dominate the requirements. Since areas grow more slowly than volumes with increasing problem size, the spatial decomposition approach, by contrast, scales both the communication costs and the computation loads.

A solution that provides both load balancing and scalable communications is the third decomposition approach, force decomposition. This concept is very new, first

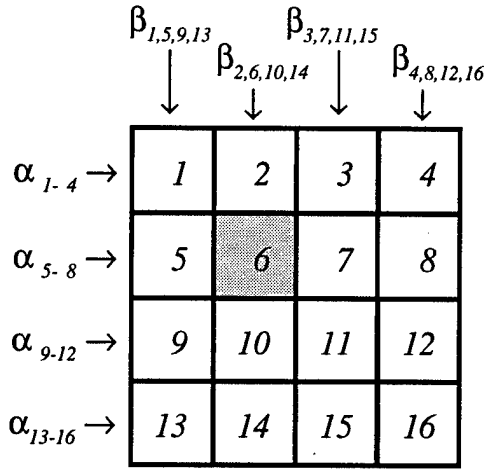


Figure 3.4. Force decomposition approach.

presented by Plimpton's 1995 article in the *Journal of Computational Physics* [32]. The approach is based on the same uniform preselection of atoms for displacement computations as found in the atom decomposition technique. The load balancing is therefore based on a similar premise. The difference lies in the fact that only a subset of atomic positions need to be communicated to each processor for each time step. This size of the subset scales down with increasing numbers of processors, so the communications scale accordingly.

Figure 3.4 shows how this technique is implemented. The computational domain is divided among $\sqrt{P} \times \sqrt{P}$ processors. Each processor contains an alpha and a beta array of atom sets. The alpha array consists of all of the atoms displaced by the processors along the same row while the beta array consists of those displaced by processors along the same column. The force computation is then performed for all of the atoms in the alpha array with the beta array used as the source for the

neighbors. Once this is performed on all of the processors, the force information is communicated and summed from the alpha processors. Since the arrays with which these other processors computed force interactions collectively contain all the atom pairs in the system, a global force computation is still successfully performed. Each processor, however, required $N/\sqrt{P}$ atoms in the alpha and beta arrays rather than the entire set of N. The additional communication cost of the forces at each step is also limited to this reduced size. The communications are therefore proportional to $3N/\sqrt{P}$. So if the number of processors exceeds nine, the cost of the communications are theoretically reduced below those required by particle decomposition.

A slightly altered approach is also presented by Plimpton in which Newton's third law allows a reduction in the computation load. This law simply states

$$\mathbf{F}_{ij} = -\mathbf{F}_{ji} \tag{3.1}$$

and reveals that a redundant operation is performed in the previous methods. Serial codes implement this law by only computing pairs where $i > j$. In the parallel techniques this would result in a severe imbalance; the first processor has only low i values while the last processor has only high i values. Instead a logic where the forces are computed for $i > j$ only if the ij sum is even, and for $i < j$ if the ij sum is odd, will retain balance and implement the law. The full atom pair contributions to the force vectors are now collected by communicating and summing from both the alpha and beta array processors. The new communication cost is $4N/\sqrt{P}$ due to the additional beta force components. The computations, however, are theoretically cut in half thereby justifying the 25% communication cost increase.

This approach was tested for the simulation of the droplet diffusion on the SP2. Linked lists and Verlet neighbor lists were still used to limit the neighbor search; they were simply applied to both the alpha and beta arrays. The force arrays, which were eliminated by the reordering of the velocity Verlet algorithm, were now required and reinstated. Additional logic steps to implement the third law computation savings were also included. The added steps reduced the efficiency such that the computation savings were only 20% instead of the desired 50% level. Also, the force communications, requiring a slightly different logic, were slower than the position

array communications. Initial results indicated that the total of these communication loads did seem to scale with increasing processors, but a disappointingly high number were required to realize the simulation time savings over the particle decomposition approach.

The resources readily available to this researcher on the SP2 did not warrant further use of the force decomposition technique. It holds promise, however, for future work on larger platforms. One caution, however, is warranted. The atom decomposition approach gains in relative scalability with increasingly complicated computation requirements; the communications remain small by comparison for increasingly larger processor sets. The project is anticipated to move in the direction of increased computational complexity (more complex molecules and chemical reactions). So the simpler, more easily modified atom decomposition technique should not be abandoned too eagerly.

3.2 Truncated Octahedron Boundaries

Cubic boundary conditions were applied during the initial development of the diffusion code. As detailed in the last chapter, there were some undesirable characteristics of this common approach. In an attempt to provide the most accurate simulation, alternate boundaries were therefore investigated. The truncated octahedron was ultimately selected as the best periodic for the simulation of the droplet diffusion. The additional boundary logic, however, threatened to slow the simulation. Fortunately, a solution was found. This section provides the details of that solution.

This more spherically shaped boundary is depicted once again in figure 3.5. The reader should note the shape fits completely within a cubic structure. Also, the hexagon planes are mathematically defined simply as

$$|x| + |y| + |z| = \frac{3}{4} L_{box} \tag{3.2}$$

where L_{box} is the length of one of the cube's sides. This means these planes are symmetrically placed within each quadrant of the three dimensional coordinate system.

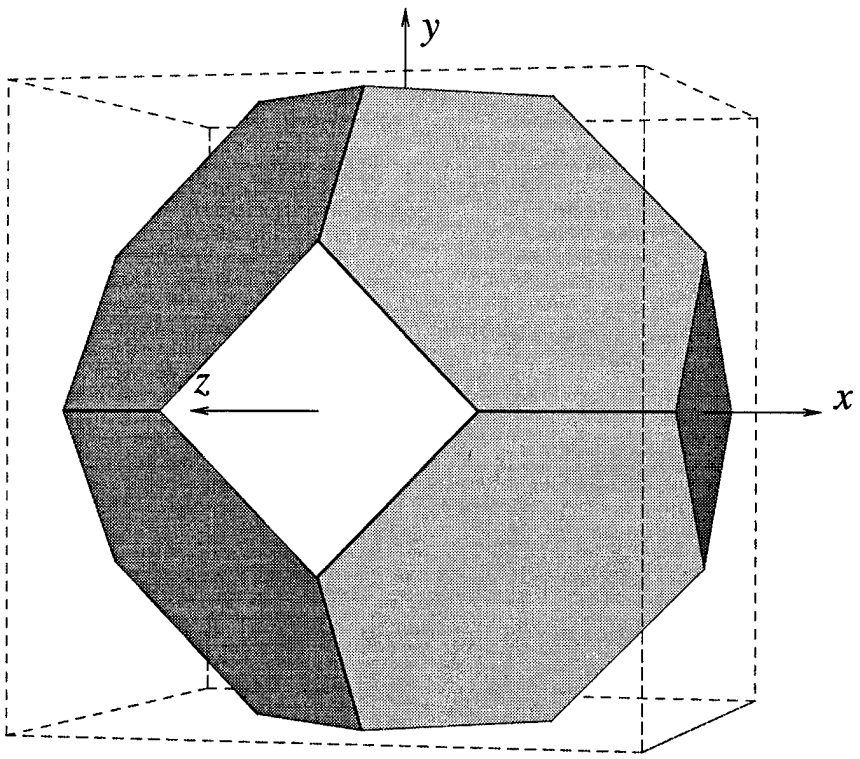


Figure 3.5. Truncated octahedron periodic boundaries.

As previously mentioned, the periodics of this geometry are set as a honeycomb structure where these planes connect cross-diagonally. All of these characteristics lead to a convenient and efficient concept for coding: a complete set of primary atoms and their images perfectly fit within the enclosing cube. Figures 3.6 and 3.7 highlight this fit by depicting the imaging of the $(-x, -y, -z)$ quadrant into the remaining half of the $(+x, +y, +z)$ quadrant. The small cubes depicted here are linked list cells. The dark, staircase shaped outer elements are cells which contain both primary and image atoms.

A simplistic approach to avoiding the image checks across the hexagon planes is to simply re-image all the primary atoms. Both the primary and image positions could be included in the linked list array since the images completely fit within the enclosing cube. This duplication of atoms would be applied to the global array; the local array would still only contain the atoms to be displaced on the local processor.

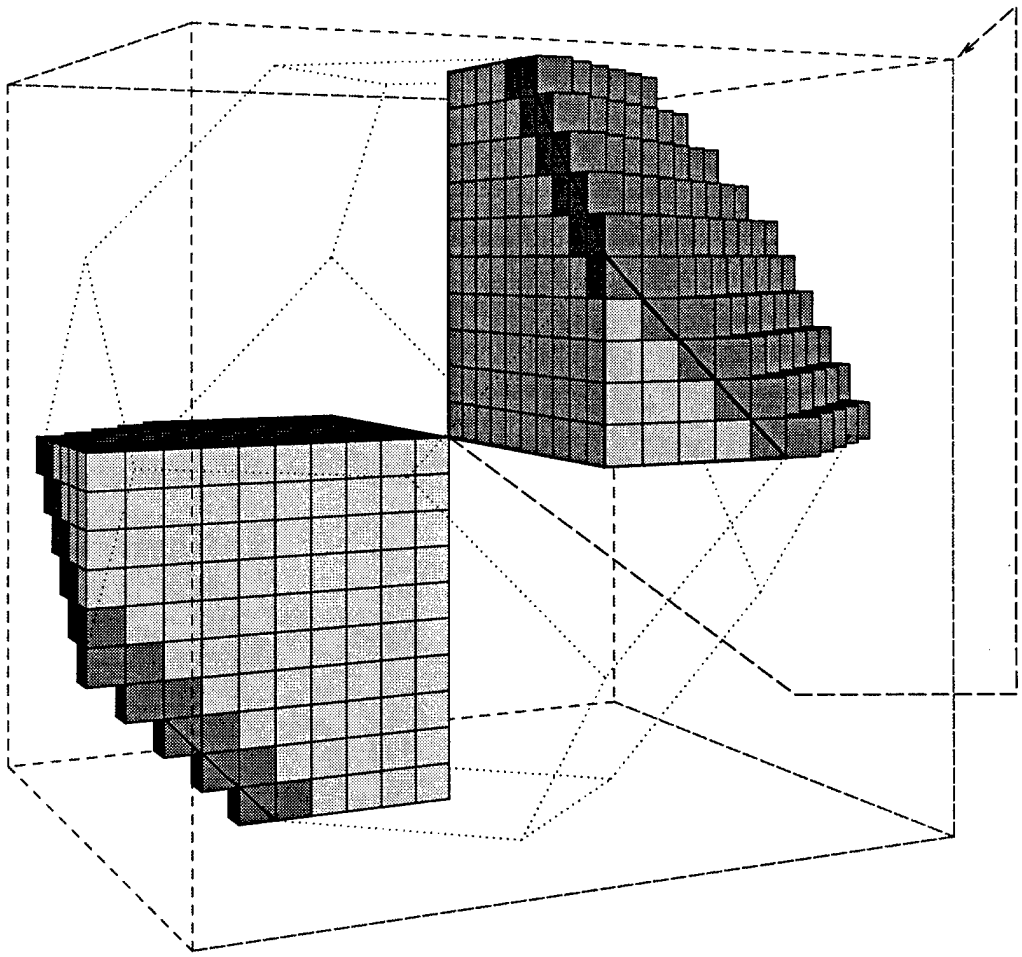


Figure 3.6. Truncated octahedron cross-quadrant periodicity.

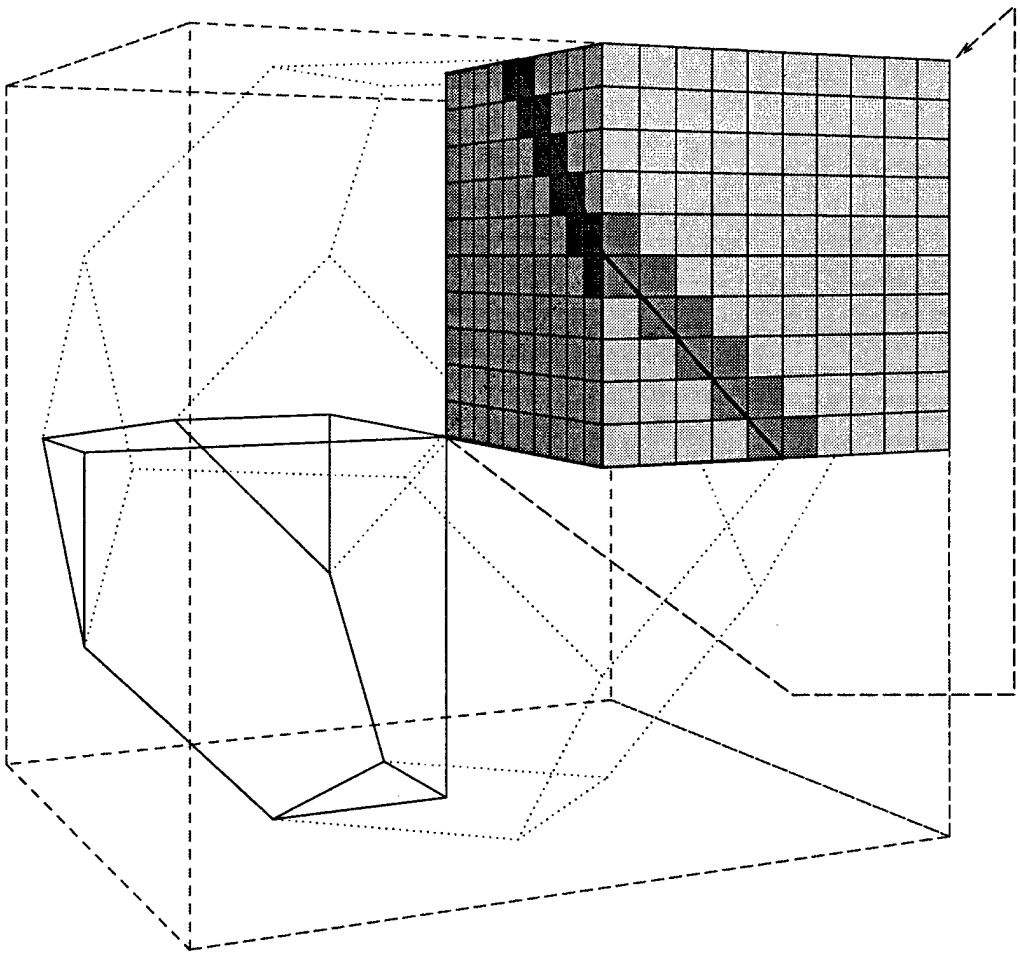


Figure 3.7. Truncated octahedron cross-quadrant meshing.

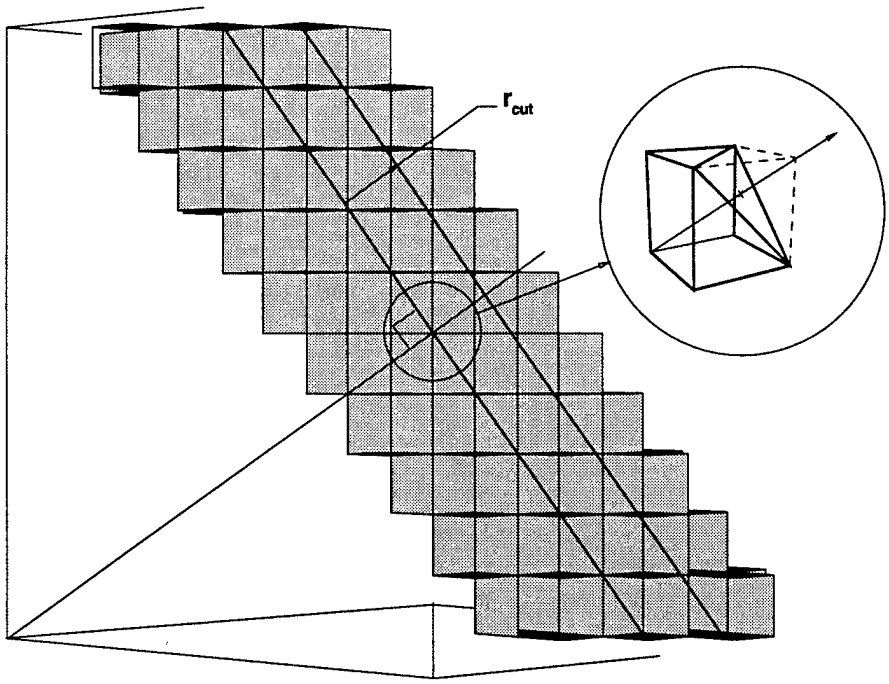


Figure 3.8. Image plane for truncated octahedron boundaries.

The computations could then proceed as before since the doubled global array would remove the required search across the hexagon planes.

The re-imaging of the complete set of atoms is certainly not necessary, however. Most of the images lie far outside any primary atom's cutoff radius. The expense of re-imaging would not only be prohibitively expensive but also very wasteful. The solution to avoiding this excessive cost is the utilizing of the linked list cell structure and presetting in a cell map those cells which contribute atomic images within the cutoff distance. This concept is depicted in figure 3.8

Reducing the imaging load down to those atoms located in the cells included in the cutoff distance is a significant benefit. There are, however, two more techniques used in the code to finely tune this enhancement. The first entails the definition of an image plane to reduce the number of image atoms down to an absolute minimum. The second is a means of inexpensively appending the image atoms onto the linked

list. Both of these techniques are relatively simple, but they are described here to clarify the complete technique.

The normal to the hexagon plane is shown in the highlight section of figure 3.8 as running cross-diagonally through the cubic structure. So the definition of the plane which exists a distance of r_{cut} inward from the boundary plane is simply

$$|x| + |y| + |z| = \frac{3}{4}L_{box} - 3L_c \tag{3.3}$$

where L_c represents the coordinate distances between the planes. Figure 3.9 shows this offset geometry and details that

$$3L_c = \sqrt{3}r_{cut} \tag{3.4}$$

Therefore, the image plane can be defined as

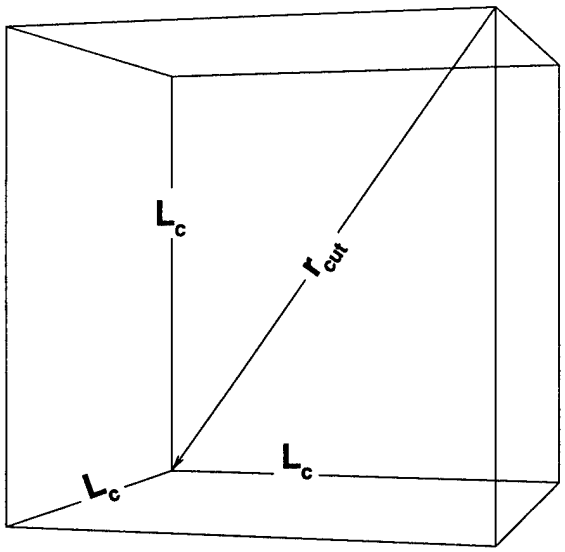
$$|x| + |y| + |z| = \frac{3}{4}L_{box} - \sqrt{3}r_{cut} \tag{3.5}$$

When the image cells are reviewed using the linked list, image candidates are checked to see if they fall outside the image plane defined using equation 3.5. Those meeting this requirement are appended as additional particles in the global array. The resultant increase in this array is now set at an efficiently minimal level.

The linked list used during the image checks is the global list since the image atoms are potential neighbors, not displacing atoms. Once the image atoms are appended to the global array, the global list must also be updated to allow boundary neighbor sorting. A simple update would involve a repeat of the linked list generation for the new global array. Much of this operation, however, would be redundant; the primary particles are already sorted into the global list structure. Fortunately, the list generating logic allows a more efficient technique.

The global linked list is generated using the following lines of code.

```
C
C     --- global linked list ---
C
      do 10 icell = 1,mcube
        ghead(icell) = 0
```



$$r_{cut} = \sqrt{3\,L_c^2}$$

therefore

$$L_c = r_{cut} / \sqrt{3}$$

and

$$3\,L_c = \sqrt{3}\; r_{cut}$$

Figure 3.9. Image plane computation.

```
10 continue
   do 20 i=1,na
     icell = 1 + int(grx(i)*rmboxinv+rmq2)
   &           + int(gry(i)*rmboxinv+rmq2)*mc
   &           + int(grz(i)*rmboxinv+rmq2)*msq
     glist(i) = ghead(icell)
     ghead(icell) = i
20 continue
```

Here the total number of atoms in the simulation is designated by the variable `na`, and `mc` represents the number of cells across one length of the cubic domain. The other variables are initialized outside the temporal loop as follows:

```
msq = mc*mc
mcube = msq*mc
rmboxinv = real(mc)*boxlength
rmq2 = real(mc)/2
```

The resultant arrays of `ghead()` and `glist()` constitute the global linked list set.

Since the lists are generated sequentially by atom number, the additional image atoms can be appended by simply continuing the linked list steps. In fact, the adding

of the image atoms to the global position arrays can be performed concurrently with the linked list updates. This process is detailed in the following lines of code.

```
C
C     --- global images ---
C
      ii = na + 1
      do 31 j=1,8,1
        do 32 n=1,icperq
          ic = (j-1)*icperq + n
          m = image(ic)
          icell = m - quad(j) + quad(9-j)
          reimage(ic) = icell
          i = ghead(m)
   33     if(i.eq.0) goto 32
            if(i.le.na) then
              x = grx(i)
              y = gry(i)
              z = grz(i)
              if((abs(x)+abs(y)+abs(z)).ge.iplane) then
                grx(ii) = x - sign(boxdq2,x)
                gry(ii) = y - sign(boxdq2,y)
                grz(ii) = z - sign(boxdq2,z)
                glist(ii) = ghead(icell)
                ghead(icell) = ii
                ii = ii + 1
              endif
            endif
            i = glist(i)
          goto 33
   32   continue
   31 continue
```

The `image()` array is the predetermined map of the cells holding potential image particles. The variable `icperq` represents the number of these cells in each quadrant. The image cell number, as a function of the primary cell location, is simply

```
icell = m - quad(j) + quad(9-j)
```

where `icell` and `m` are the image and primary cell numbers respectively, and the `quad(j)` and `quad(9-j)` variables are the first cell numbers in each respective quadrant. This information is saved in a small cell map, `reimage()`, for use during later

image updates. As shown, the global position vector is compared to the previously determined image plane value in accordance with equation 3.5. If the image check is met, the global position array and the linked lists are both appended accordingly. The only additional logic required here is the starting of the image particle numbers at one greater than the total simulation value and the checking of image candidates to ensure they are not already images. This is required since the global link list is dynamically updated and will contain image candidates as the appending progresses.

These techniques, therefore, efficiently set the global array as both the entire set of atomic positions and the required image positions across the hexagon planes. During a force computation, the only cells requiring boundary checks are the side facing cells. The order of the cell map used during the simulation is depicted in figure 3.10 (only the first quadrant is shown for clarity). The side facing cells are reserved as the last cells in the array. So up to the first call of a side cell, all of the force computations are performed without boundary condition complications. The force loop cost of this enhanced boundary is therefore greatly offset.

The enhancements incorporated into the code with this more appropriate boundary resulted in a fairly efficient tool. The truncated octahedron code was timed at about a five percent slower rate than a similar cubic code with the same number of atoms. Two considerations, however, made the new approach the logical choice. As mentioned in the previous chapter, 48% of the cubic boundary volume falls outside the largest included sphere. By comparison, only 32% of the truncated octahedron volume falls in this undesirable category. Also, the effects of periodic images on the diffusion model are significantly reduced by the new conditions as evidenced by moving the image ratio 60% closer to the optimum spherical value. So a five percent penalty on the modeling of the environment is easily offset by the required volume reduction and the enhanced simulation physics.

One final point is warranted before closing this section. These timing comparisons were performed on codes which incorporated linked list search techniques only; the Verlet neighbor lists had not been added. When the beneficial effects of the combined approach were realized, the Verlet list was desired as part of the truncated octahedron code. At first this appeared difficult. The Verlet list purposefully avoids

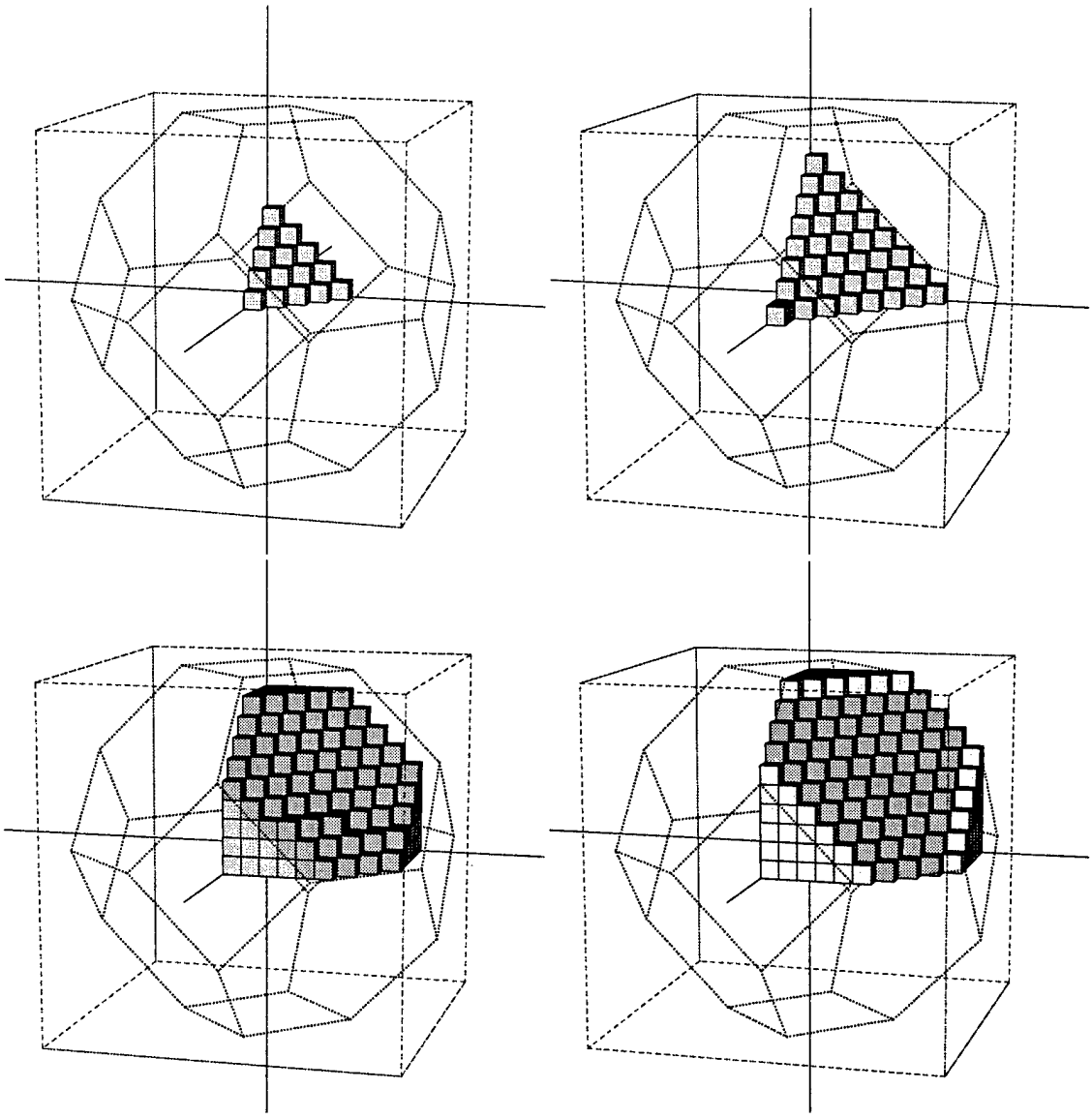


Figure 3.10. The side facing cells are saved as the final elements.

updating the linked list to achieve greater neighbor searching efficiency. But the identification of image particles was set as part of this update. The images still needed to be properly represented between list updates to accurately model the boundaries.

A very easy solution, implemented in the existing code, consists of simply delaying the reimage process until a linked list update is required. The image plane is defined based on the list radius instead of the cutoff radius to validate the use of a fixed set of image candidates during neighbor list loops. Also, the position of the image atoms during these steps are updated from the associated primary atomic positions. The identification of these global position elements is set during the image checks in yet another small pointer array. The updating of the image coordinates is therefore very efficient. So this simple solution allowed the inclusion of the Verlet list into the truncated octahedron code. Chapter five will show that this resultant code competes very well with existing cubic based molecular dynamic codes.

3.3 Property Computations

The evaluation of atomic motions during the simulation runs proved insightful but still too limiting to answer many of the questions associated with supercritical diffusion. In an attempt to track the physics of the process, an array of thermodynamic properties were computed from the atomic data. A means of measuring the radial distribution function was modified from standard techniques to allow dynamic tracking of the droplet interior structure. Density and temperature measures based on established theory were also performed. A surface tension measure, however, was uniquely developed to fill the desire to track this important droplet property. This section details the computations employed to produce these properties. Also, using the linked list cell structure, an informative contour plot display of the data is reviewed as the close of the chapter.

3.3.1 Radial Distributions

The radial distribution function results from the evaluation of the diffractions of X-rays, neutrons or electrons during experimental studies. The shape of this function is distinctly different for the three phases. Tabor [6] describes the function as the

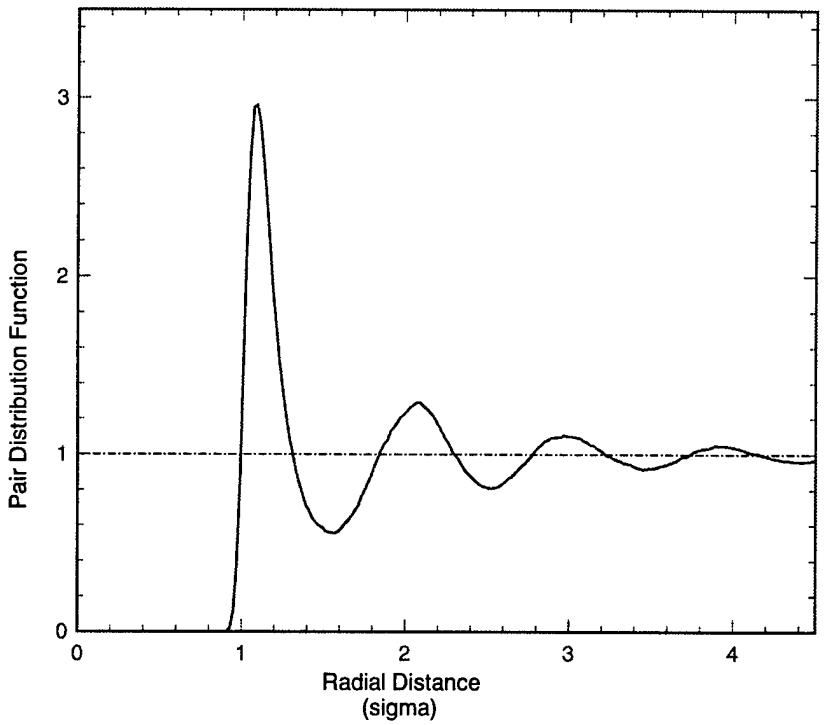


Figure 3.11. Pair distribution function for argon. Plotted from simulated liquid structure at 85 K and 2.14×10^{28} atoms/m^3.

average number of molecules present in constant volume spherical shells emanating from a randomly selected molecule. This can be measured in molecular dynamic simulations. The method consists of averaging the densities based on pair separation distances and dividing by the average density of the entire system. An example of the resultant plot, termed the pair distribution function, is shown in figure 3.11.

This is a typical shape for a liquid and matches the experimentally derived radial distribution. The peaks are located where higher density regions are found in the low energy wells. A solid structure would display constant value spikes at these locations due to the continuous lattice structure. Even a gas shows a weak collection at the first peak, but the random nature eliminates the remaining spikes. The liquid, as shown, displays a short-range order over several atomic diameters. Beyond this distance, the function approaches unity indicating a random order similar to a gas. So a display of the pair distribution helps to determine the phase of the system.

Allen and Tildesley detail the coding which allows the computation of the pair distribution for a bulk liquid simulation. First the number of particles which fall within a spherical shell gap, `delr`, around another particle are saved in an array termed `hist()`.

```
      do 100 i=1,N-1
        do 99 j=i+1,N

            ... calculate minimum image distances ...
            ...        (rxij,ryij,rzij)           ...

          rijsq = rxij*rxij + ryij*ryij + rzij*rzij
          rij = sqrt(rijsq)
          bin = int(rij/delr) + 1
          if(bin.le.maxbin) then
            hist(bin) = hist(bin) + 2
          endif
   99   continue
  100   continue
```

The variable `maxbin` is used to ensure the distribution is not performed beyond the periodicity of the simulation.

The actual pair distribution function can then be computed using the `hist()` array. Since the array includes pair data from all the atoms, it actually represents an average over N particles. Also, the loops above can be performed over numerous time steps to help expand the statistical average. In the code below, this value is termed `nstep`.

```
      const = 4.0*pi*rho/3.0
      do 10 bin=1,maxbin
        rlower = real(bin-1)*delr
        rupper = rlower + delr
        nideal = const*(rupper**3 - rlower**3)
        gr(bin) = real(hist(bin)) / real(N*nstep) / nideal
   10 continue
```

Here the pair distribution is normalized by the number of atoms which would exist in the spherical shell defined by `rlower` and `rupper` if the density were equal to the average system density. This procedure sets the previously mentioned asymptotic trend toward unity as the order of the system becomes essentially random.

The system comprised of the evaporating droplet and its surroundings is certainly not homogeneous, but it is very large. Attempting to use the standard approach to computing the pair distribution would be very slow (an order N^2 computation) and would provide an average of the dual phases of no practical use. A simple adjustment, however, provides a means of efficiently computing the distribution for only the center of the droplet. Instead of looping over all atom pairs, only the atoms within a core set of cells are used. The linked list structure is used to find the atoms within a requested set of cells. This is a very easy procedure since the cell map emanates from the center of the simulation. The pairs are determined by looping through the surrounding 125 cells (allowing a separation distance of about five atomic diameters). So the computations are no longer N^2.

This modified code is shown below

```
      do 102 im = 1,mcore
        mdp = gocell(im)
        k = head(mdp)
101     if(k.eq.0) goto 102
          x = rx(k)
          y = ry(k)
          z = rz(k)
          i = nskip + k
          ntau = ntau + 1.d0
          do 103 ic = -2,2,1
            do 104 jc = -2,2,1
              do 105 kc = -2,2,1
                m = mdp + ic + jc*mc + kc*msq
                j = ghead(m)
106             if(j.eq.0) goto 105
                if(i.eq.j) goto 107
                  rxij = x - grx(j)
                  ryij = y - gry(j)
                  rzij = z - grz(j)
                  rsq = rxij**2 + ryij**2 + rzij**2
                  rij = sqrt(rsq)
                  bin = int(rij/delr) + 1
                  if(bin.le.maxbin) then
                    hist(bin) = hist(bin) + 1
                  endif
107               j = glist(j)
                goto 106
105           continue
```

```
104        continue
103      continue
       k = list(k)
       goto 101
102 continue
```

Other than the linked list sorting, only minor changes are required. The adding of one to the `hist()` array instead of two is one of these. This is required since the ij pairs are redundantly determined in this new procedure. The additional savings of parallel computations greatly outweigh this limitation. Also, minimum image lengths are ignored since the core is assumed to be significantly far removed from the boundaries. The `maxbin` variable is retained to limit the search to a predetermined radius (this is limited, as just mentioned, to $2 \times cell_{length}$ by the 125 cell search pattern). Finally, the normalizing density must now be a value determined from the core set of cells, not the entire system. This is easily performed by utilizing the pair counts and the known core volume. Chapter five will show that this technique proved very valuable in determining the phase structure present in the droplet interior during the very dynamic diffusion process.

3.3.2 Temperature

The measurement of the temperature profile during the simulation was also based on a standard approach. Allen and Tildesley, again, provide a description, but this can be found in many other molecular dynamic texts as well. The temperature is simply a function of the kinetic energy of the atoms in a system. An instantaneous kinetic temperature, $\mathcal{T}$ is defined by Allen and Tildesley for monatomic systems as

$$\mathcal{T} = 2\mathcal{K}/3Nk_b = \frac{1}{3Nk_b}\sum_{i=1}^{N} m_i {v_i}^2 \tag{3.6}$$

where $\mathcal{K}$ is the kinetic energy of the entire set of atoms, N, and k_b is the Boltzmann constant. This function is time-averaged to determine the temperature of bulk systems.

A bulk average is output by the code developed herein along with the kinetic and potential energy profiles. Table 3.1 is an example of the tracking. This is of

minor importance, however. The more informative measures of temperature involve utilizing the linked list cells once again as small system domains. A running tracking of the droplet core and the surrounding environment temperatures are output by presetting a collection of core and environment cells. Also, saving each cell's averages allows the development of a three dimensional thermal contour. Both of these display formats are discussed at the close of this section.

3.3.3 Density

The density profiling utilizes a technique first utilized by Thompson et al. [21] and further refined by Maruyama [22]. It is quite simple and also very efficient. The number of neighbors found within the cutoff sphere for each atom are tabulated and represent what is termed herein as a local atomic density. This value can be compared to an atomic number density by simply dividing by the volume of the cutoff sphere. The efficiency of the technique is obvious; the cutoff neighbors must already be determined as part of the inter-atomic force computations.

The original work of Thompson et al. used the technique to define an equilibrated droplet cluster. The atoms which had a local density greater than a set average between saturated liquid and vapor values were defined as droplet atoms. Maruyama extended the concept to define the actual surface of his equilibrated droplets by comparing the local densities to a narrow band centered between the saturated points. While both of these works were the genesis of the utilization herein, the concept is expanded further. The definition of an equilibrated droplet entering a supercritical environment and the setting of cell based data sets are both dependent on the technique. The droplet initialization will be discussed in the next chapter, and the contour profile development will be reserved as the closing of this chapter. The core and environment region computations are, however, presented at this time.

Table 3.2 shows the output format of the previously mentioned core and environment regions. As shown, both the temperature and density values were averaged over these domains. Additionally, a curve-fit based on molecular dynamic data from Nicolas et al. [43] allows the computation of the environment pressure as a function of the density and temperature. This data was collected over a set value of time

Temperature and Energy Tracking
...and elapsed time (minutes) and list updates between info writes.

Iter#	T(K)	Kinetic	Potential	Total	Time	List Updates
0	170.6	0.6874D-16				
500	170.7	0.6879D-16	-.6082D-16	0.7970D-17	1.116	53
1000	169.5	0.6830D-16	-.6053D-16	0.7769D-17	1.136	52
1500	168.9	0.6804D-16	-.5968D-16	0.8364D-17	1.174	53
2000	170.0	0.6851D-16	-.5923D-16	0.9283D-17	1.185	52
2500	171.2	0.6897D-16	-.5859D-16	0.1038D-16	1.205	54
3000	170.7	0.6877D-16	-.5777D-16	0.1101D-16	1.207	54
3500	171.5	0.6911D-16	-.5738D-16	0.1173D-16	1.214	52
4000	171.6	0.6915D-16	-.5692D-16	0.1224D-16	1.217	51
4500	170.2	0.6858D-16	-.5598D-16	0.1260D-16	1.228	54
5000	169.3	0.6822D-16	-.5536D-16	0.1286D-16	1.224	52
5500	169.6	0.6832D-16	-.5491D-16	0.1341D-16	1.241	53
6000	168.8	0.6801D-16	-.5449D-16	0.1352D-16	1.236	52
6500	168.8	0.6801D-16	-.5407D-16	0.1394D-16	1.253	51
7000	169.2	0.6817D-16	-.5358D-16	0.1459D-16	1.286	53
7500	169.2	0.6819D-16	-.5317D-16	0.1502D-16	1.240	52
8000	169.3	0.6820D-16	-.5272D-16	0.1548D-16	1.240	52
8500	168.8	0.6803D-16	-.5245D-16	0.1558D-16	1.249	53
9000	168.9	0.6807D-16	-.5209D-16	0.1598D-16	1.264	56
9500	170.0	0.6850D-16	-.5186D-16	0.1664D-16	1.259	54
10000	170.8	0.6882D-16	-.5153D-16	0.1729D-16	1.264	55

Table 3.1. Bulk temperature and energy tracking output.

steps prior to each write. Typically fifty time steps were utilized to allow a statistical averaging without smoothing over the simulation dynamics. Plots generated from this information are presented in chapter five.

The reader may note the absence of the droplet interior pressure values. The dynamic measure in this high density region proved unsatisfactory. This caused some initial concern over the ability to dynamically measure the surface tension. Classical approaches utilize the pressure differential for this measure. Fortunately a unique method was devised.

3.3.4 Surface Tension

The surface tension computation is based on the molecular interpretation presented by Tabor [6]. The following is the excerpt that led to the development.

> The free surface energy of a liquid lends itself to a very simple molecular interpretation. Molecules in the bulk are subjected to attraction by surrounding molecules; the field is symmetrical and has no net effect. At the surface, however, the surface molecules are pulled in towards the bulk of the liquid. Apart from a few vapor molecules there is no attraction in the opposite direction. Consequently if we wish to increase the area we have to pull molecules up to the surface from the bulk against this one-sided attraction. This accounts for the surface energy.

This concept is shown pictorially in figure 3.12. The attraction forces within the interior of the drop are greater due to the increased density and average proximity of neighboring atoms. The accumulation of the force vectors, however cancel the force effect in both the drop and the environment; only a surface, where a density gradient exists, will generate a net force. The pulling against this force when expanding the surface is the energy described by Tabor.

The structure of a drop results directly from the minimizing of this energy. Since the forces exist only on the droplet surface, their sum is lowest for the shape with a minimal surface area per volume. This shape is the sphere. If a drop is exposed to relatively low density surroundings (such as a subcritical environment), the one-sided forces remain and the spherical structure is retained. When exposed to a high density region (such as a supercritical environment), however, the attractive forces

Property Computation Results

	Core (32 cells)		Environment (872 cells)		
iter#	Density (atoms/m^3)	Temperature (K)	Density (atoms/m^3)	Temperature (K)	Pressure (MPa)
500	1.8469D+28	102.96	3.6444D+27	198.83	7.5575
1000	1.8640D+28	100.73	3.4248D+27	191.13	6.7386
1500	1.8306D+28	99.94	3.4259D+27	193.01	6.8575
2000	1.8040D+28	103.30	3.3944D+27	198.21	7.1314
2500	1.7714D+28	108.83	3.4458D+27	195.80	7.0611
3000	1.7454D+28	107.70	3.3273D+27	191.87	6.6456
3500	1.7669D+28	117.40	3.3655D+27	194.87	6.8831
4000	1.6988D+28	111.31	3.3646D+27	194.03	6.8303
4500	1.6271D+28	110.25	3.4361D+27	186.75	6.4793
5000	1.6482D+28	116.91	3.4516D+27	190.53	6.7384
5500	1.6048D+28	119.92	3.4404D+27	190.69	6.7327
6000	1.5899D+28	120.97	3.4344D+27	184.70	6.3486
6500	1.5788D+28	120.61	3.5056D+27	186.41	6.5468
7000	1.5004D+28	117.85	3.4909D+27	190.15	6.7678
7500	1.5400D+28	120.91	3.5399D+27	186.94	6.6248
8000	1.4580D+28	116.33	3.5684D+27	182.64	6.3772
8500	1.4729D+28	123.65	3.6050D+27	184.71	6.5581
9000	1.5182D+28	128.60	3.5788D+27	183.52	6.4474
9500	1.4603D+28	129.54	3.6863D+27	187.28	6.8308
10000	1.4497D+28	125.76	3.6312D+27	186.26	6.6940

Table 3.2. Represented in tabular form for a collection of core and environment cells.

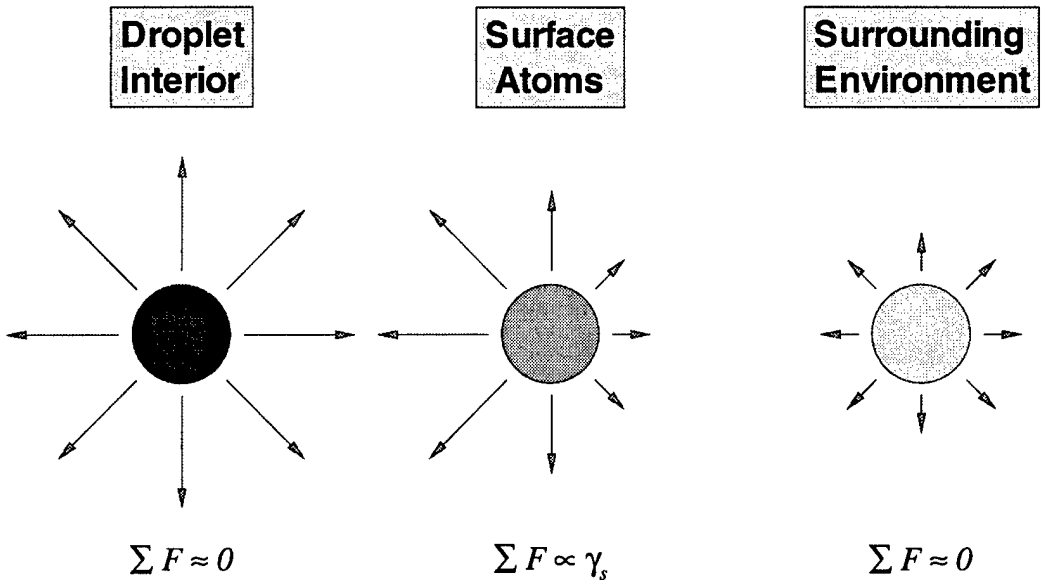


Figure 3.12. The tension on the surface of a droplet is proportional to the imbalanced attraction force. Atoms in the dense droplet interior and the lower density surroundings experience offsetting forces.

outside the drop become significant. This reduces the one-sided nature of the forces at the surface. The droplet surface energy is then very low and the spherical nature of the drop less likely to be retained. Additionally, the low surface energy results in surface elements which are much more likely to be broken free of the drop when hit by high speed atoms from the surroundings. This is why supercritical evaporation is so effective.

Focusing back on the means of measuring this important droplet property, the measuring of forces on each atom is already performed in the simulation. The simple description by Tabor seems to imply that each atom's forces could be tabulated and presented as the surface tension. As with the other properties, this would result in a statistically insufficient technique. Instead the sum of the force vectors on the atoms

are maintained for each cell. At the end of the time averaging period, the average force on each atom in this cell can be computed simply as

$$F_{i_{avg}} = \sqrt{\left(\frac{\langle f_{i_x}\rangle}{N_{icell}}\right)^2 + \left(\frac{\langle f_{i_y}\rangle}{N_{icell}}\right)^2 + \left(\frac{\langle f_{i_z}\rangle}{N_{icell}}\right)^2} \tag{3.7}$$

The N_{icell} value represents a summation of all atoms found in the cell over the specified time averaging period. Therefore, the time and atomic averaging is performed concurrently when dividing by this single value.

The importance of tabulating the force components as opposed to just the interatomic force magnitudes cannot be overemphasized. The seemingly random atomic interactions do have a directional nature on the surface. The computing of this effect requires both magnitude and direction. The summing of just magnitudes results in an apparent surface tension throughout the liquid structure, but the surface tension in a bulk fluid is known to be inconsequential. The cancelling of the large forces only occurs when the directional nature of the interactions are retained. This same directional nature gives the surface cells significantly larger force results by comparison.

One more important modification must be addressed before moving on. Steele [44] noted that the implementation of the force tabulation method would probably fail due to the significance of the repulsive forces. The surface tension is an indication of the one-sided forces due to attraction imbalances, but repulsive forces can easily dominate a force summation. Cell averaging of the data diminishes this problem. High level collision forces, equal and opposite in direction between the associated atoms, cancel when summed within the same cell. During the tabulation, however, a collision which occurs at a cell boundary (such that the forces are tabulated within different cells) will still overshadow the relatively weak attraction forces. To counter this problem, the code computes the forces using the following approach.

```
      rsqinv = 1.d0/rsq
      sqr2 = sigsq*rsqinv
      sqr6 = sqr2*sqr2*sqr2
      sqr12 = sqr6*sqr6
      fqr = (2.d0*sqr12-sqr6)*rsqinv
      if(rsq.lt.rcsqa) then
C         ...repulsive force
```

```
          fxr = fxr + fqr*rxij
          fyr = fyr + fqr*ryij
          fzr = fzr + fqr*rzij
        else
C           ...attractive force
          fxa = fxa + fqr*rxij
          fya = fya + fqr*ryij
          fza = fza + fqr*rzij
        endif
```

The filtering variable, `rcsqa`, is set during the initialization of the code as $(\sqrt[6]{(2\sigma)})^2$. This is determined by setting the force to zero in the force computation (equation 2.6).

The forces required for atomic translation computations are then determined using

```
fx = fxa + fxr
fy = fya + fyr
fz = fza + fzr
```

The filtered set of attraction only forces are available, though, for the surface tension tabulations when desired.

So, to summarize the technique, the attraction force components on each atom are summed for all atoms in each linked-list cell. Along with the summing of these forces, the cell based number of contributing atoms to this sum is recorded. If requested, this sum can be continued over several time steps to provide time averaging of this data. After the requested time period is complete, the force components are averaged by dividing the sums by the number of contributions. This now gives the average force vector on each atom in the cell. Within bulk regions, these values approach zero, but on the droplet surface, the one sided forces are readily apparent.

3.3.5 Contour Displays

The representation of this surface tension force in a tabular form was not very informative. The sum total of the system surface tensions did drop during the diffusion process, but a more detailed visualization of the effect was desired. This need helped push the development of a volumetric contour representation of the data. This

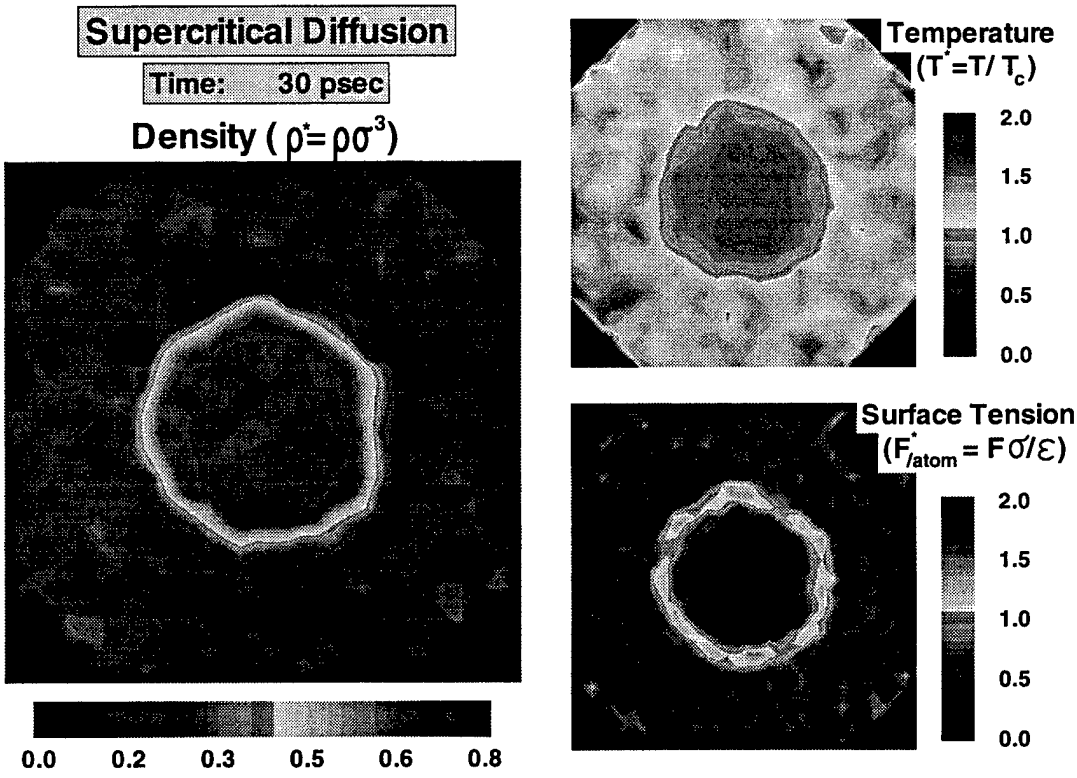


Figure 3.13. A representation of density, temperature and surface tension central contours during a simulation of 27,000 atoms evaporating into a 200 K and 7.5 MPa supercritical environment.

proved very beneficial not only for the display of the surface tension, but also in the dynamic tracking of the temperature and density profiles.

The representation of all three properties is shown in figure 3.13. The depiction shows a central slice of the beginning of a 91,000 atom simulation (27,000 atom drop evaporating into a 64,000 atom environment). The value of such a tool will become evident in Chapter 5 when an assortment of simulations are reviewed. The remainder of this section will detail the code output and post processing that generate these contour images.

The evaporation code outputs cell based data at requested times during the simulations. The data consists of six property values per cell collected over a specified number of time-steps prior to the output process. This allows both spatial and

temporal averaging of the data. The properties collected are the local atomic density, $\langle \rho_{icell} \rangle$, the velocities squared, $\langle v^2{}_{icell} \rangle$, the attractive force components, $\langle f_{xicell} \rangle$, $\langle f_{yicell} \rangle$, and $\langle f_{zicell} \rangle$, and the count of the number of atoms associated with each of these accumulations, N_{icell}. The brackets, $\langle\ \rangle$, are used to indicate that each of these properties is actually a cell-based sum over both time and space. Before detailing the computations which convert these properties into the inputs for the contours shown in figure 3.13, a data smoothing technique is reviewed.

Each of the cell-based values consist of many samples for the cells located in the liquid structure, but significantly less in the environment. This is simply a result of fewer atoms in the low density regions contributing to the statistical averaging. To compensate for the resultant noise which occurs in the contours each cell can be represented as a *supercell* as follows

$$\langle p_{icell} \rangle = \langle p_{icell} \rangle + w_p \sum_{jcell=1}^{26} \langle p_{jcell} \rangle \tag{3.8}$$

$$N_{icell} = N_{icell} + w_p \sum_{jcell=1}^{26} N_{jcell} \tag{3.9}$$

where $\langle p_{icell} \rangle$ represents one of the collected data variables and N_{icell} the associated accumulation count. The variable w_p is a weighting factor. If this factor is set to 0, then equations 3.8 and 3.9 leave the accumulation sets unchanged. If, however, a factor of 1 is used, the computations set each cell-based variable to a super-set of data from the primary cell and the surrounding 26 cells. Weights in between allow the information from the primary cell to attain greater significance during the averaging. The closer the weighting factor gets to 0, the more dominant the primary cell data becomes.

This smoothing process is applied to the simulation output as a post processing computation. Each of the variables can be smoothed independently. Typically weighting values of 10% to 20% applied to the velocity and force field values yield good images for the temperature and tension data respectively. The density profiles are already somewhat smoothed by the nature of the cutoff sphere tabulation and therefore do not typically use this technique.

The conversion from the collected cell-based values to contour property values is fairly straightforward. The density values represent a summation of the local atomic densities for every sample. So an average value for the contour is simply

$$\rho_{icell} = \frac{\langle \rho_{icell} \rangle}{N_{icell}} \tag{3.10}$$

Similarly, the velocity squares represent a summation which is directly proportional to the kinetic energy found in each cell over all the samples. Using the kinetic temperature function (equation 3.6), the average temperature of each cell is therefore

$$\mathcal{T}_{icell} = \frac{m\langle v^2{}_{icell} \rangle}{3N_{icell}k_b} \tag{3.11}$$

The surface tension measure has already been developed in the previous section in equation 3.7. The supercell data is applied directly as cell-based data.

As shown, the truncated octahedron code has been developed to be efficient and provide useful information. Before detailing the results of several simulations using this code, a review of the techniques utilized to initialize the simulation systems is provided. This is the basis of the next chapter.

Chapter 4

SIMULATION DESCRIPTION

As noted in Chapter 1, the scope of the research presented herein is limited to the evaluation of the associated physics of an equilibrated droplet suddenly exposed to a quiescent, high temperature and high pressure environment. The initialization of a simulation to achieve the goals of this research proved challenging. Standard molecular dynamic approaches to set equilibrated systems were readily available. But an established method of starting the unique two phase system of a droplet diffusing into highly energetic surroundings did not exist. Initial attempts yielded insightful but unsatisfactory results. This chapter details the evolution of a series of techniques which proved quite successful.

4.1 Initialization

Both Haile [35] and Allen and Tildesley [36] outline a fairly universal standard to initialize liquid and gas structures for molecular dynamics. A face-centered cubic lattice is established, a Gaussian distribution is applied to initialize the velocities, and the simulation is allowed to run to an equilibrated state. During this run the lattice 'melts' into a physically real structure, but care must be taken to ensure the desired state point is reached. To better understand the melting process, a review of the potential function is warranted.

The Lennard-Jones potential shown in Chapter 2 (figure 2.2) and the radial distribution function discussed in Chapter 3 both indicate a natural collecting of atoms at the energy well location. A sparse lattice initially has very little potential interactions. The relaxing to a real system results in a collection of atoms tending to reside in the potential well, at least momentarily. Since total energy remains constant in a closed system, the resultant increase in the negative value of the sum of the potentials can only occur if the kinetic energy correspondingly increases. By contrast, a dense lattice initially sets the inter-atomic spacing such that the sum negative

potential energy is rather high. The relaxing of the lattice results in momentary substructures of solid-like particles. The potentials within these groups are significant, but not between them. The net effect is a reduction in the potential effect, or an increase in the potential energy. Now the conservation of energy requires that the kinetic energy drop. So, in short, the system kinetic energy will rise when a sparse lattice equilibrates, but will drop during the relaxing of a dense lattice.

So the initial temperature, directly proportional to the kinetic energy, will shift during equilibration. Determining the magnitude and direction of the shift a priori is difficult and unnecessary. A thermal control technique is more easily incorporated. This is the approach used in all the initializing codes to compensate for the melting effects. It enables the setting of fairly precise state points.

This control alone, however, is insufficient to initialize system. Attempts at setting the two phases of the droplet and supercritical environments concurrently in the same computational domain proved untenable. While a relaxing of the lattice structures of the two phases certainly occurred, equilibrium could not be established in the presence of the inter-phase thermal and density gradients. Even when assuming quasi-equilibrium states, the required fine tuning to establish the desired phases was overwhelmed by the dynamic exchanges. As a result, two independent equilibrations are utilized: one for the droplet and another for the surrounding environment.

4.1.1 Environment

The term *environment* refers to the surroundings into which the droplet evaporates during the simulation. Since the environment is simply a bulk system, the establishing of this initial structure closely follows standard approaches. This section details these methods, including the previously mentioned thermal control. The control enables the setting of the environment at a desired initial thermodynamic state.

The face-centered cubic (F.C.C.) lattice is universally recommended as the initial structure for molecular dynamic simulations. This is established by setting a series of sub lattice sets of four atoms at evenly spaced points (see figure 4.1). The periodicity is automatically provided in cubic simulations with this technique by using a spacing

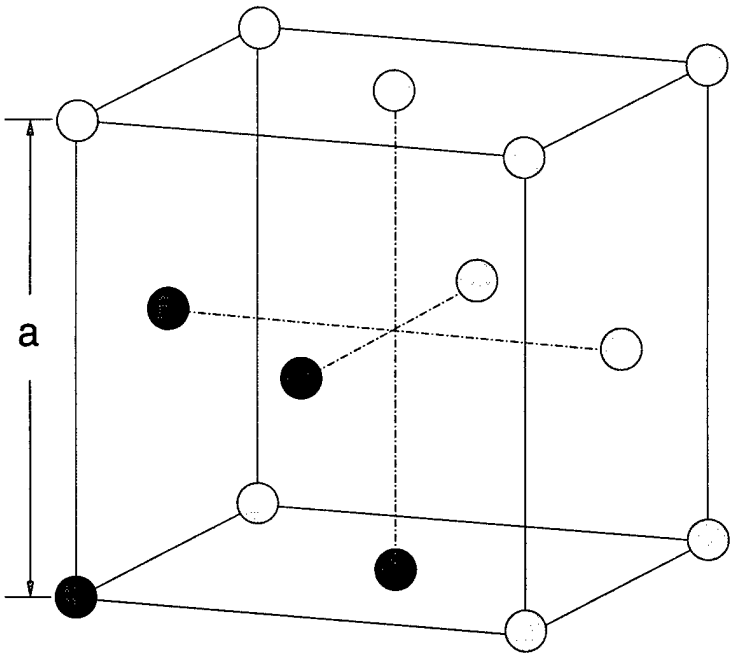


Figure 4.1. Face-centered cubic structure. The sub-lattice of the four shaded atoms can be used to generate the full lattice.

evenly divisible into the domain length. Using the truncated octahedron boundaries, this same periodicity is set by filling only the left half of the simulation cube. Half the cube length is correspondingly set to be evenly divisible by the spacing. The octahedron structure is established by re-imaging atoms where appropriate into the right side of the simulation. This concept is depicted in figure 4.2.

Two more points should be noted before proceeding. The full lattice structure contains $4I^3$ particles, where I is an integer value. The resultant density of this structure is $4/a^3$, where a is the length of the lattice spacing. These factors make attaining a precise density level for a desired domain size difficult. This is overcome in the code by initializing the full lattice and then uniformally erasing atoms as necessary to attain the desired density. One more lattice adjustment recommended by Schofield [45] is also utilized. The melting process is further enhanced by shifting each particle slightly from the lattice positions. The adjustment is performed unevenly (using a

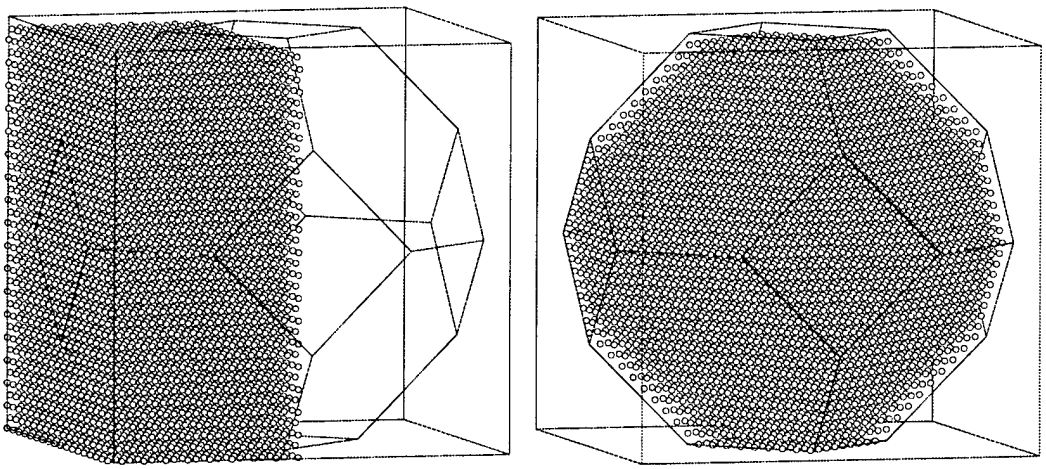


Figure 4.2. Initializing the truncated octahedron lattice. The left half of the F.C.C. lattice in the containing cube is re-imaged to ensure proper periodics.

random number generator from Press et al. [46]) and therefore presets a desired randomness into the simulation.

The velocities of equilibrated systems can be set in a Maxwellian distribution [35].

$$f(v_x)dv_x = \frac{N(v_x)dv_x}{N} = \sqrt{\frac{m}{2\pi k_b T}} \exp(-m{v_x}^2/2k_bT)dv_x \tag{4.1}$$

This is a Gaussian about zero with a specific standard deviation of

$$\sigma = \sqrt{k_bT/m} \tag{4.2}$$

Utilizing a Gaussian subroutine from Press et al. [46], the code assigns the velocity components in accordance with this distribution. Before proceeding with the equilibration procedure, however, a system drift correction is performed.

The Gaussian routine should set an average velocity of zero. This is desired to establish the quiescent environments. Since random numbers are utilized in the Gaussian, this is not a certainty. With the large simulation sizes investigated herein, the

statistical averaging admittedly results in very small system drifts. But the correcting procedure is quite simple, so the velocity components are reset using

$$v_{ix}{}^{new} = v_{ix}{}^{old} - \frac{1}{N}\sum_{i}^{N} v_{ix}{}^{old} \tag{4.3}$$

$$v_{iy}{}^{new} = v_{iy}{}^{old} - \frac{1}{N}\sum_{i}^{N} v_{iy}{}^{old} \tag{4.4}$$

$$v_{iz}{}^{new} = v_{iz}{}^{old} - \frac{1}{N}\sum_{i}^{N} v_{iz}{}^{old} \tag{4.5}$$

The environment simulation is now ready to run. The atoms are set into their lattice positions and the Maxwellian velocity distribution established. As mentioned, simply running to equilibration will result in an unpredictable property setting. A thermal control is therefore applied during the beginning of the process. There are many sophisticated methods to run constant temperature simulations. A very basic technique from Allen and Tildesley [36] is used instead. The purpose of the control is to simply counter the kinetic energy shift due to the relaxing of the lattice structure. The control is not needed for the duration of the simulation. It is removed once the lattice melting is complete. Any imperfections in the simulation are quickly smoothed thereafter.

The control is based on adjusting the kinetic energy after each time step to a level associated with the desired temperature. The components of the velocity are regulated using

$$v_{ix}{}^{new} = \sqrt{\frac{3\mathcal{T}_{des}k_bN}{2\mathcal{K}_{old}}}\, v_{ix}{}^{old} \tag{4.6}$$

$$v_{iy}{}^{new} = \sqrt{\frac{3\mathcal{T}_{des}k_bN}{2\mathcal{K}_{old}}}\, v_{iy}{}^{old} \tag{4.7}$$

$$v_{iz}{}^{new} = \sqrt{\frac{3\mathcal{T}_{des}k_bN}{2\mathcal{K}_{old}}}\, v_{iz}{}^{old} \tag{4.8}$$

where $\mathcal{K}_{old}$ is the pre-adjusted system kinetic energy. The temperature is now set to the desired level, $\mathcal{T}_{des}$, for the next time step. The validity of this statement can be shown by noting from the kinetic temperature function (equation 3.6)

$$\mathcal{T}_{new} = \frac{m}{3Nk_b}\sum_{i=1}^{N}[({v_{ix}}^{new})^2 + ({v_{iy}}^{new})^2 + ({v_{iz}}^{new})^2] \tag{4.9}$$

Substituting equations 4.6, 4.7, and 4.8 into 4.9 yields

$$\mathcal{T}_{new} = \frac{m}{3Nk_b}\,\frac{3\mathcal{T}_{des}k_bN}{2\mathcal{K}_{old}}\,\frac{2\mathcal{K}_{old}}{m} = \mathcal{T}_{des} \tag{4.10}$$

The thermal adjustment is therefore performed at very little computational cost. The component multiplier is simply a fixed constant times a tabulated total kinetic energy value. This is applied uniformally throughout the velocity field at the end of each time step. Also, the control logic automatically adjusts the temperature up or down as required. So a directional determination of the relaxing of the lattice is not necessary.

The results of an equilibration run for an environment domain are shown in figure 4.3. This shows the tracking of the system energy levels through 10,000 time steps. Thermal control is applied during the first 5,000 steps and then removed. The kinetic energy, and therefore the system temperature as well, remains steady throughout the run. The control ensures this at first, and then the equilibrated state point is evident in the remainder of the plot. The change in the potential energy indicates the melting process of the lattice. Since the initial structure is sparse, the potential energy drops. Without the control this would be transferred into kinetic energy. This particular system would equilibrate to 245K instead of the desired 200K as a result.

The reader may note from figure 4.3 that the 5,000 step control period is overly cautious. The procedure to initialize the droplet structure uses a similarly careful approach. The next section details the associated techniques. Simulation results are also shown, and they reveal the extended control period is no longer so cautious for this case.

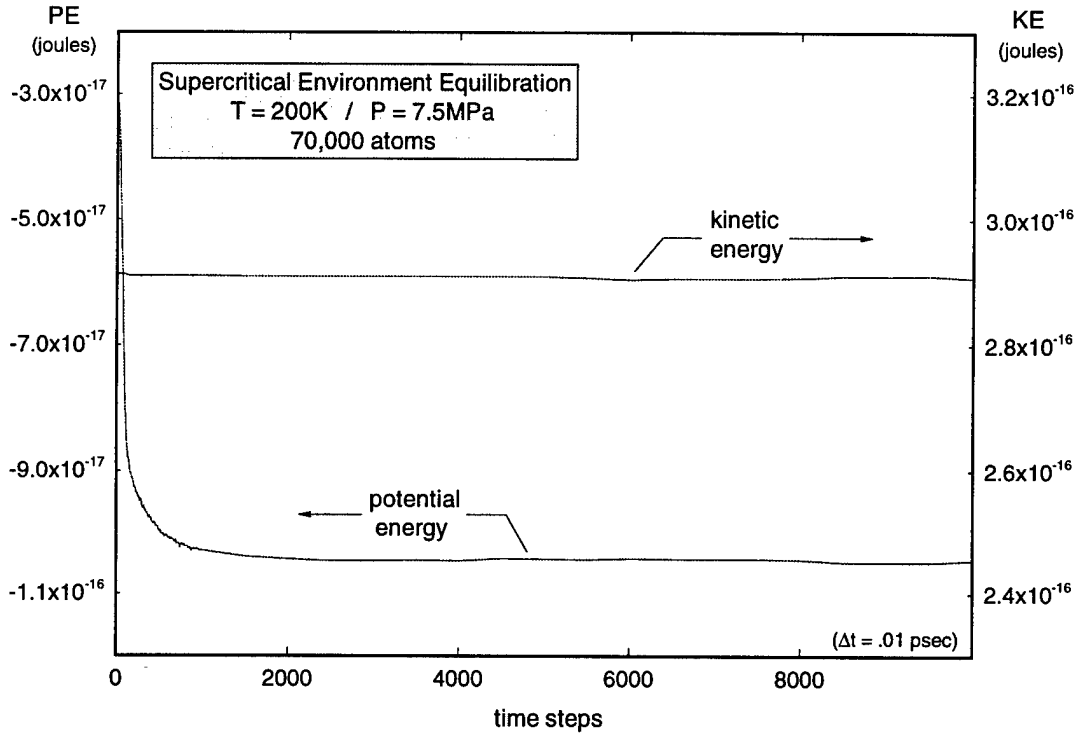


Figure 4.3. Environment equilibration. Thermal control removed at time step 5000.

4.1.2 Droplet

The initial concepts for establishing a droplet structure were based on equilibrating a bulk liquid structure and then fusing the results into the supercritical environment. The surface tension of the liquid would theoretically form the spherical droplet structure relatively quickly during the beginning of the diffusion process. This failed. When the equilibrated liquid structure was placed into the lower density environment, the density gradient provided the desired surface tension. But this tension was not present in the equilibrating phase of the bulk liquid because the boundaries were periodic. The fused liquid quickly collapsed under the sudden appearance of the inward directed forces. The simulation still progressed, but the core pair distribution function and temperature tracking showed an initial meta-stable state. In other

words, rather than simulating the diffusion of a liquid drop, the code was initially modeling the sublimation of a nearly solid structure.

So a new approach was necessary. Literature from Thompson et al.[21] and Maruyama [22] covering equilibrated droplet investigations provided the necessary insight. They established their systems by placing the initial lattice structure into a larger open domain. The definition of the vapor pressure states that when a liquid is exposed to a vacuum, evaporation will occur until this rate equals the rate of condensation. The pressure exerted by the surrounding vapor at this equilibrium condition is the saturation vapor pressure [6]. When the initial lattice structures are placed in the larger domains, they likewise evaporate until surrounded by the equilibrated vapor phase. In addition, the inwardly directed surface forces cause the lattice to form into a minimum energy shape, the sphere. The surrounding domain must be large enough to avoid boundary effects on the droplet structure, but not so large as to require significant evaporation (this would be very time consuming).

The concept is successfully employed for the truncated octahedron initializing code by simply setting a gap between the lattice and the bounding periodic walls. With this space properly set, the simulation results in a droplet floating in saturated vapor (see figure 4.4). Actually, acknowledging the periodics, the model is an infinite number of such droplets. An occasional re-centering of the center of mass of the system maintains the droplet structures at the center of each computational domain. The focus of this discussion will therefore remain on only one such image.

The initial thermal control in this simulation is very important; without it the droplet would collapse and freeze into a meta-stable solid-like structure. This effect is not only due to the previously discussed lattice melting effect, but also due to the latent heat of vaporization. When the atoms in the drop separate due to evaporation, the absolute level of potential energy drops. But this means the actual value of the potential increases toward the zero level. Since, once again, energy must be conserved in a closed system, this shifting of the potentials can only occur as a result of the kinetic energy dropping. So the temperature of the system will correspondingly drop as the diffusion progresses to equilibrium.

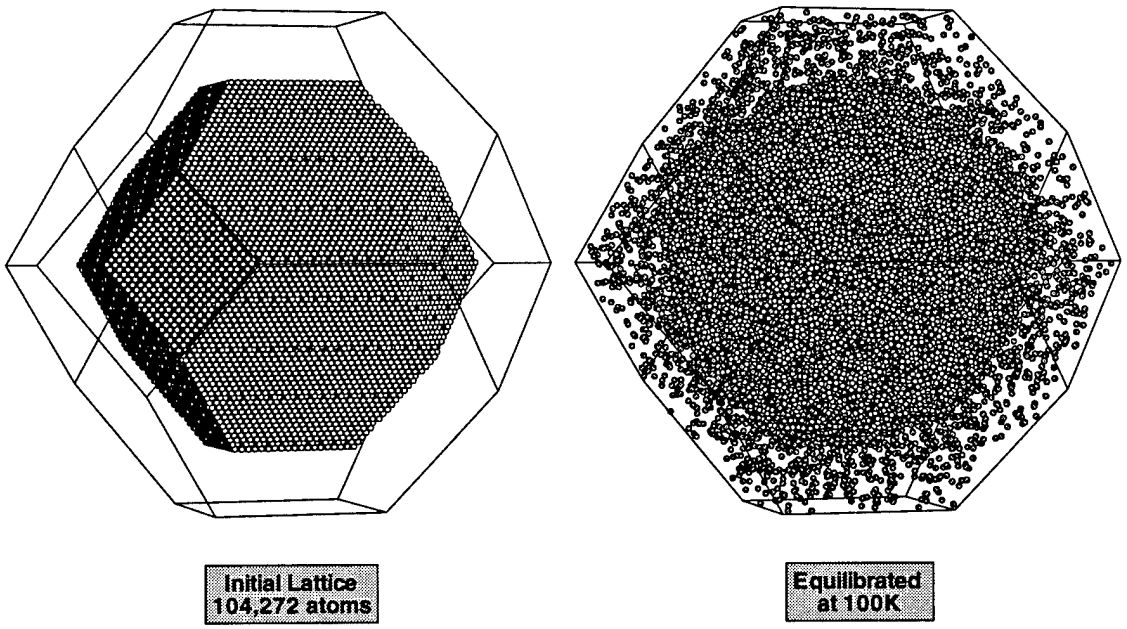


Figure 4.4. Droplet initial lattice and equilibrated state.

Figures 4.5, 4.6 and 4.7 reveal the relative success of the thermal control in establishing an equilibrated drop. The plots present energy, density and thermal profiles respectively for a 7,300 atom system. Unlike the overly cautious bulk environment runs, the droplets must be simulated long enough to establish the previously mentioned vapor pressure. Figure 4.6 reveals that this is achieved only after 15,000 time steps have elapsed. Also, the extended thermal control is required to compensate for the latent heat. Figure 4.5 shows that even after 5,000 time steps, the kinetic energy drops slightly due to this effect. The impact on the droplet temperature past the control point is not significant, however, as shown in the thermal profile of figure 4.7. Interestingly, this figure also shows that the thermal control actually super-heats the vapor phase. This effect diminishes, however, as the vapor population increases and vanishes after the control is removed.

Figure 4.6 also shows the liquid core and surrounding vapor densities settling at apparently respective ends of the saturated spectrum. These values should be fairly

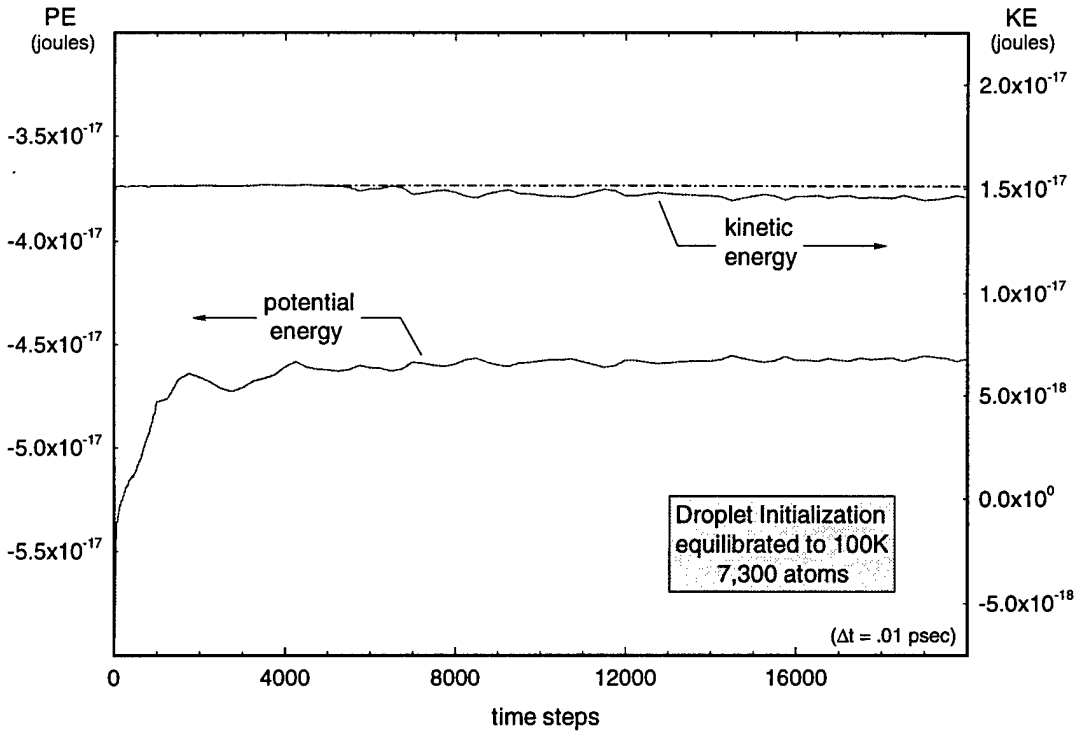


Figure 4.5. Energy profile of a droplet equilibration. The thermal control is removed at 5000 steps. The slight drop in the kinetic energy thereafter is due to the latent heat effect.

close to saturated vapor and liquid levels at 100K. Comparisons to values tabulated in Vasserman et al. [2] show that the droplet core is 5% of the saturated density span from the liquid level, while the environment reading is only a 2% shift from saturated vapor. These results provide confidence in both the equilibration procedure and the chosen values of the Lennard-Jones parameters used in the potential functions.

4.2 Fusing of Droplet into Environment

The equilibrating of the droplet and the surrounding environment at desired state points does not complete the necessary initialization process. The two systems must be fused together. The proper approach to handle this challenge proved initially elusive. Fortunately a combination of the previously defined local atomic density in conjunction with a geometrical viewing formula yields satisfactory results. Before

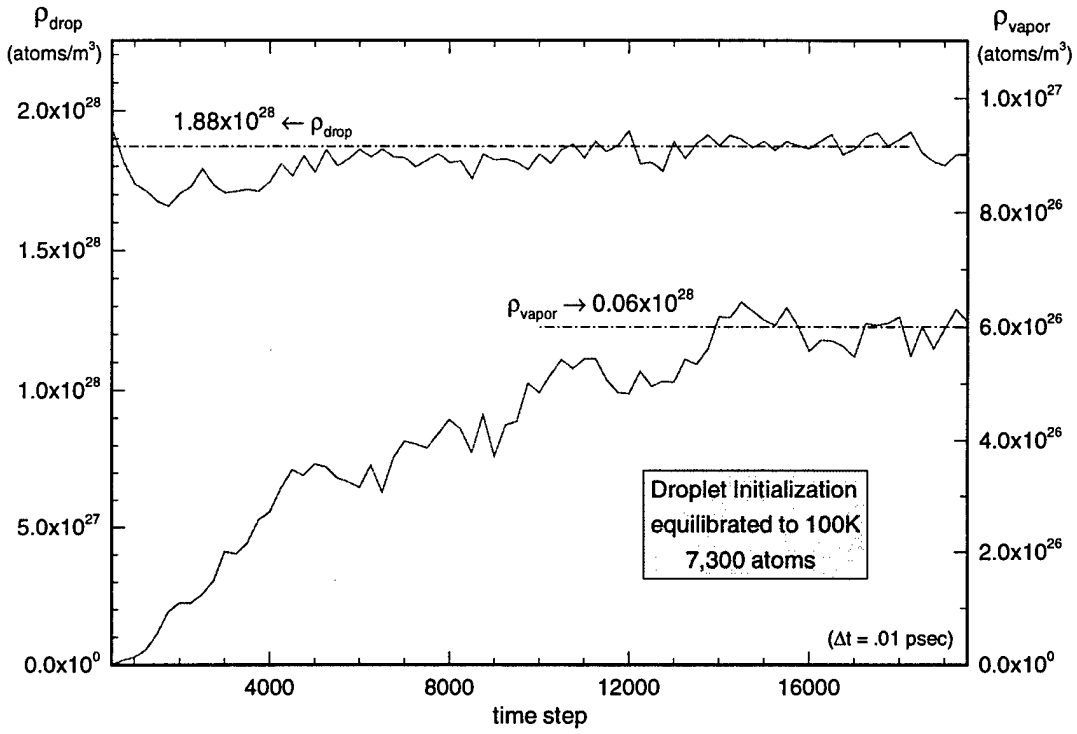


Figure 4.6. Density profile of a droplet equilibration. The establishment of the equilibrated vapor pressure is indicated by the progression and leveling of the surrounding vapor density.

detailing this technique, however, a short review of the lesser approaches is provided for comparative purposes.

The first attempt involved simply placing the entire droplet simulation into an equivalently sized space in the center of the environment. The environment atoms in this region were removed to provide the opening. This was fairly successful, but the saturated vapor density was lower than the surrounding supercritical density. This created what was termed a *halo* effect of low density between the drop and the environment. The rapid dynamics of the simulation caused concern that this region could effect the initial and critical stages of the diffusion process. To correct this, the droplet was assumed spherical and only those atoms within the assumed radius were fused. The visualization of the surface tension, however, revealed a flaw. The

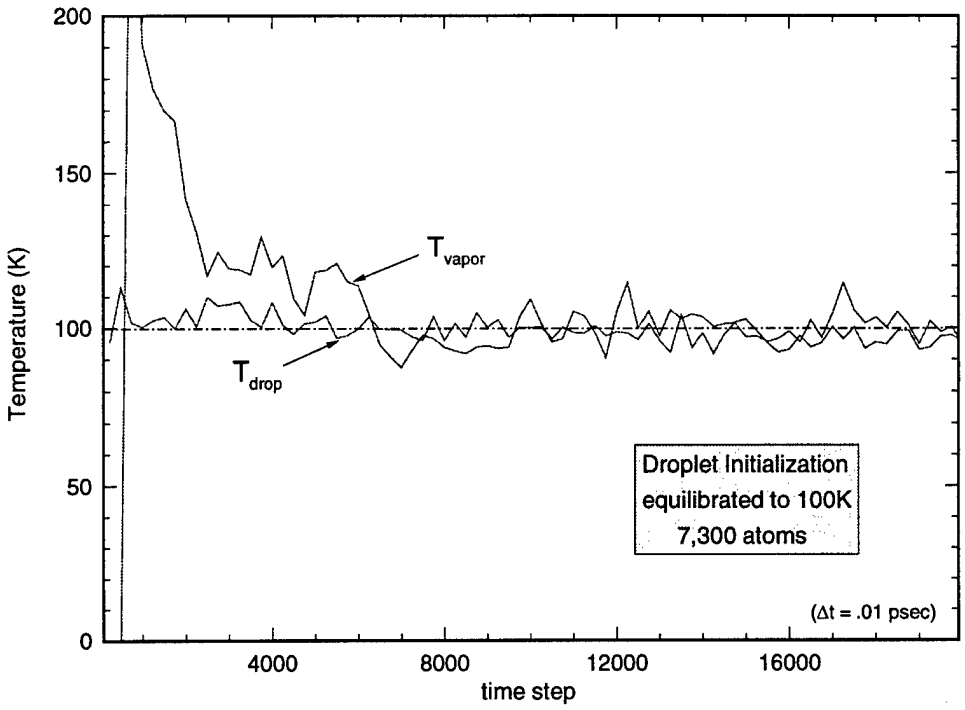


Figure 4.7. Thermal profile of a droplet equilibration. The vapor phase is super-heated by the thermal control.

technique cut into the surface and improperly distorted the surface energy. A means of fusing the actual shape of the droplet into the environment was therefore desired.

The means of defining the actual droplet was found in Thompson's Oxford studies [21]. The local atomic density is used to filter out those atoms that fall below a set level. A value of 0.35×10^{28} atoms/m^2 has proven quite useful as a cutoff to remove the vapor phase from the equilibrated droplet. Once this filtering is complete, the drop is ready to be placed into the environment. Before this can occur, however, an opening must be formed in the environment.

The reader should realize that the equilibrated droplet shape based on local density filtering is not a perfect sphere. When filtering the droplet atoms, however, the radius of the outermost particle can be retained. When fusing the two systems, any environment atom which exists outside this outermost droplet particle radius can

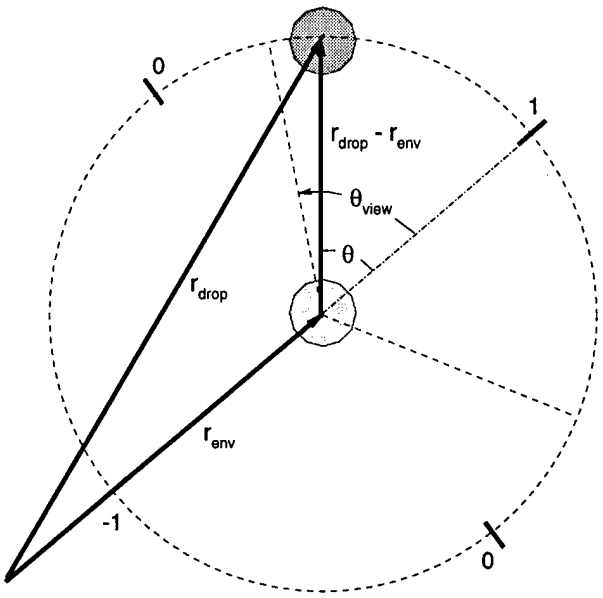


Figure 4.8. Fusing geometry. If the environment atom 'sees' a droplet atom within a preset view angle, the atom is erased.

be automatically retained; they fall outside the droplet structure. If the remaining environment atoms could be made to geometrically view their surroundings then the problem is solved. Any environment atom, when looking outward from the center of the simulation, which 'sees' a liquid atom can be assumed to reside within the intended droplet shape. Such atoms would be removed to form the required fuse space. The cosine law from basic trigonometry provides the necessary logic.

Figure 4.8 shows an environment atom 'looking' at a nearby atom from the fused droplet. The cosine law states

$$\mathbf{r}_{env} \cdot (\mathbf{r}_{drop} - \mathbf{r}_{env}) = |\mathbf{r}_{env}| \, |(\mathbf{r}_{drop} - \mathbf{r}_{env})| \, \cos(\theta) \tag{4.11}$$

So the coordinates of the environment and the droplet atom can be used to determine the view angle with respect to the line emanating from the origin to the environment atom.

$$r_{env}^2 = x_{env}^2 + y_{env}^2 + z_{env}^2 \tag{4.12}$$

$$\cos(\theta) = \frac{(x_{env}x_{drop} + y_{env}y_{drop} + z_{env}z_{drop}) - r_{env}^2}{\sqrt{r_{env}^2[(x_{drop} - x_{env})^2 + (y_{drop} - y_{env})^2 + (z_{drop} - z_{env})^2]}} \tag{4.13}$$

The values surrounding the circle on figure 4.8 are the resultant $\cos(\theta)$ values at the respective locations. As shown, this value attains a maximum of one at a view angle of 0° and decreases uniformally to negative one for a view angle of 180°. So if

$$\cos(\theta) \geq \cos(\theta_{view}) \tag{4.14}$$

where θ_{view} is a preset view angle, then the droplet atom is within the 'view' of the environment atom. If this occurs, the environment atom is assumed to be located inside the liquid structure of the droplet and erased. If, however, all of the droplet atoms within the neighborhood of this atom are outside of the view angle, then the environment atom is retained.

The fusing process is performed by a post processing routine. Appendix B is a listing of this code. As just detailed, the droplet is defined first by filtering the vapor phase atoms using the local density comparisons. A second pass removes atoms which lie outside of the droplet but were previously part of a dense group in the vapor phase. The environment atoms are then checked for overlapping with the droplet. If they lie outside the maximum droplet atom position, they are retained and the next atom reviewed. If they fall within the potential droplet region, a linked list of the droplet atoms is utilized to limit the liquid atom view checks to those within two cell lengths. This is a straight-forward review, periodic boundary complications are not present since the droplet is significantly far from the environment boundaries. Once all of the environment atoms are reviewed the fusing is complete.

At the end of this procedure, the code allows the writing of the droplet and environment atoms into a Tecplot file for visualization [1] . The atoms are sorted by category of whether they were retained or not. Figure 4.9 shows a central slice of data from an actual procedure. The left-most image is of the environment atoms prior to the view checking. The gap shown in the right-most image results from the viewing process. The image in the center shows the placing of the filtered droplet structure into the open space. As shown, the droplet fits quite nicely.

The simulations using this fusing technique as an initializing step provide the desired initial model. There is a short period of adjustment required due to the removing of atoms from each simulation, but this appears to occur quickly and without undue consequences. These observations can be seen in figure 3.13 at the end of Chapter 3. The surface tension is clearly retained in a uniform fashion, and the halo effect in the density profile is not present. The diffusion process is therefore ready to proceed.

4.3 Diffusion

The diffusion simulation has been discussed in some detail in the preceding chapters of this report. A modeling of the diffusion of a saturated droplet into a variety of surrounding environments is desired. The combined linked list and Verlet neighbor list search techniques significantly reduce the workload of the force computations. Also, the truncated octahedron periodic boundaries are utilized to reduce the boundary effects and enhance the efficiency of the environmental volume. Particle decomposition based parallel coding allows load balanced modeling, an important capability for the two phase system. Finally, a cell aware structure and cell based outputs yield insightful property data in both tabulated and contour formats. There is one more concept, however, that should be covered. The latent heat of evaporation required during the long diffusion simulation seriously erodes the driving potential of the high temperature environment. This section details the approach utilized to overcome this effect.

[1] Tecplot is a registered trademark for a data visualization program by Amtec Engineering, Inc. Version 6 was used for the imaging performed in support of this thesis.

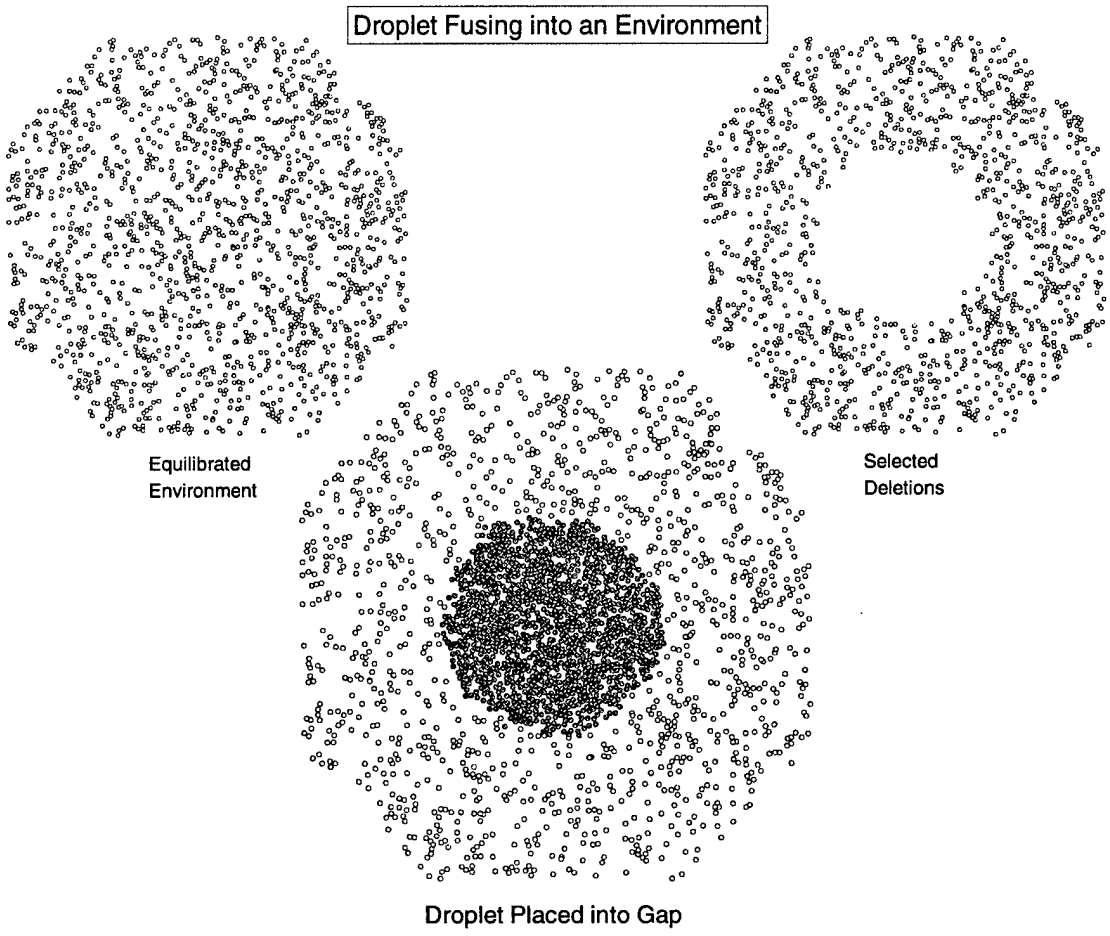


Figure 4.9. Droplet fusing into an equilibrated environment. Central slice of particles shown here.

The technique is actually quite simple, but appears to work well. The thermal control applied to the systems during the initializing runs is selectively utilized throughout the diffusion run as well. The latent heat effect was discussed previously in the detailing of the droplet initializations. This is simply the absorption of kinetic energy due to the relaxation of the droplet structure. The resultant reduction in the thermal driving potential can therefore be averted by bumping the system energy gradually upward.

The reader may note at this point that a thermal control of the entire system would seem to induce a non-physical diffusion model. The resulting simulation output would be simply a function of this control. The author agrees. Instead the thermal control is applied only to a fully encasing volume located at the computational boundary (see figure 4.10). The velocity components of the atoms which exist within this zone are adjusted at each time step to maintain the original environment temperature. The rest of the simulation is left to run freely with only the inter-atomic potentials and momentum transfers influencing the model. In other words, a thermal heat source is established at a significant enough distance from the diffusion dynamics to retain the simulation physics. This source also, however, provides the necessary energy to sustain the driving potential of the hot surroundings.

The coding required to establish this selective heating is performed by taking advantage of the atomic image array already established. This array defines the atoms which lie at a distance of r_{cut} from the hexagon boundaries. This encompasses most of the desired thermal control set; only the atoms in the side facing cells remain unspecified. The cell map order allows these particles to be easily appended since the final elements in the map are these side cells. Once the array of controlled particles is determined, their velocities are scaled using the same logic as before

$$v_{ix}{}^{new} = \sqrt{\frac{3\mathcal{T}_{env}k_bN_{heated}}{2\mathcal{K}_{old}}}\, v_{ix}{}^{old} \tag{4.15}$$

$$v_{iy}{}^{new} = \sqrt{\frac{3\mathcal{T}_{env}k_bN_{heated}}{2\mathcal{K}_{old}}}\, v_{iy}{}^{old} \tag{4.16}$$

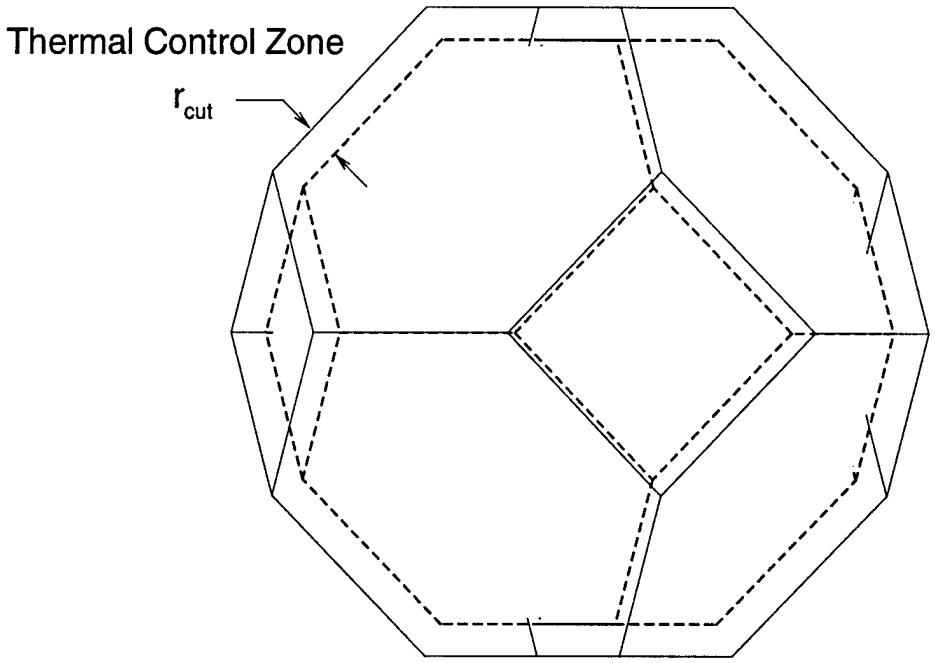


Figure 4.10. Thermal control zone. Set as an r_{cut} thick volume measured inward from the boundary.

$$v_{iz}{}^{new} = \sqrt{\frac{3T_{env}k_bN_{heated}}{2\mathcal{K}_{old}}}\, v_{iz}{}^{old} \tag{4.17}$$

Now, however, $\mathcal{K}_{old}$ is the total kinetic energy of this select group of N_{heated} atoms, and T_{env} is the temperature of the original surrounding environment. These adjustments are applied at every step to sustain the thermal condition and to limit any resultant computational instabilities.

The reader should now be familiar with the assortment of both standard and unique approaches utilized to provide a viable simulation of the supercritical diffusion process. The simulation of a droplet diffusing into assorted high temperature and pressure environments is initialized by independently modeling the two phases. This separation allows the use of careful controls to reach desired conditions for both. The fusing of the two simulations is performed in a manner to retain the surface integrity of the drop. The technique also exposes this surface immediately to the diffusion driving environment. The thermal blanket surrounding the domain boundaries counters

the dampening effects of the heat of vaporization. This allows a complete diffusion simulation. Quite interesting results were obtained when running these carefully devised models of both subcritical and supercritical processes. The next chapter details these results.

Chapter 5

RESULTS AND DISCUSSION

This thesis is intended to provide initial insight into the supercritical diffusion process through the development and use of a molecular dynamic tool. The previous chapters detailed the generation of the tool. An assortment of two subcritical and two supercritical environment conditions were simulated to provide the desired insight. Figure 5.1, a copy of figure 1.1 from Vasserman [2] and Reynolds [3], shows these four simulation points. These are the environmental conditions which drive the diffusion of the droplet. The droplet initial conditions are also shown on the figure along the saturated liquid line at a pressure of .32 MPa. The subcritical cases, running in regimes which are already well understood, were performed to provide validation of the simulation. The droplets for these two and the two supercritical runs were all initialized to a temperature of 100 K and contained approximately 5,600 atoms. The supercritical run at 200 K and 7.5 MPa was also repeated for drops of 27,000 and 100,000 initial atoms. These runs reveal encouraging scaling possibilities. This chapter provides details and numerous visualizations concerning all of these cases. Before these details are presented, however, this chapter begins with a review of code validation and performance comparisons.

5.1 Code Performance

The code utilized to support the research was a continually evolving tool. The validation and performance measures presented here are results from the most recent version of the code. This includes truncated octahedron boundary conditions with cell aware coding, linked lists, and Verlet neighbor lists. The validation is limited to a review of the simulation algorithm's ability to properly model a bulk liquid simulation. This is measured by comparing a pair distribution plot generated by the code to results provided by Allen and Tildesley [36]. Further validation of the

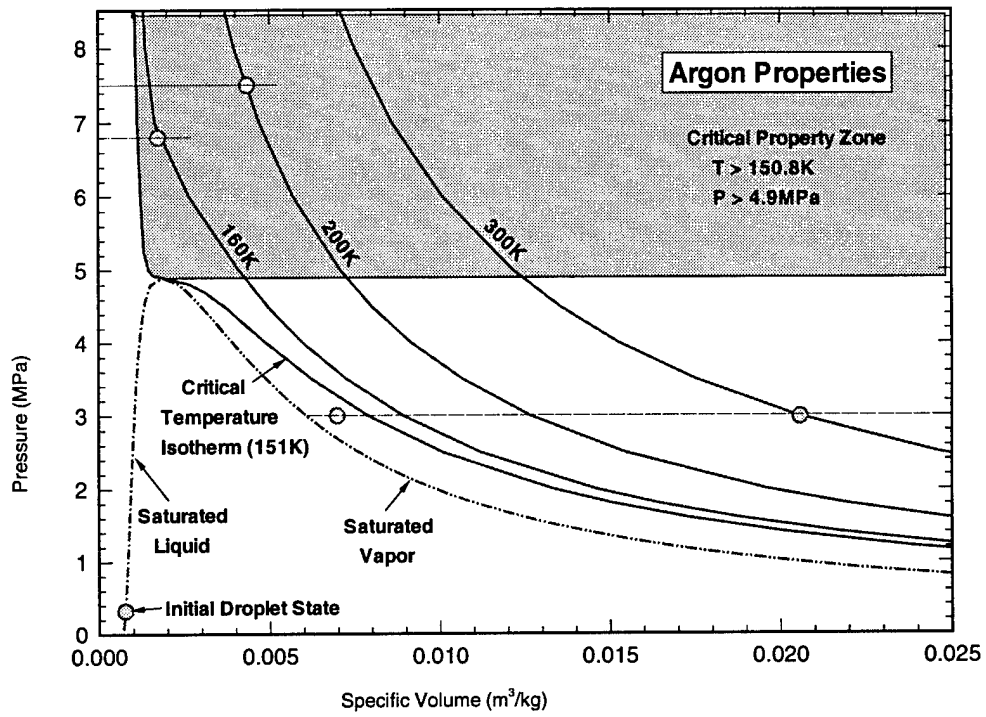


Figure 5.1. Simulation Conditions. The four gas phase points show the range of simulated environments. All of the drops were set at saturated conditions at 100 K (and therefore .32 MPa) [2] [3].

simulation process is provided by the subcritical run results. The follow-on section details these.

The performance measures begin with a series of comparisons to runs performed by Plimpton [32]. He provides a detailed array of performance measures from a cubic simulation on both serial and parallel platforms. Since his measures are based on simulations of bulk liquid argon, the performance comparisons detailed herein are likewise limited to bulk liquid simulations. The section ends with a discussion of the parallel performance of the code when running the actual diffusion simulation. The success of the load balancing particle decomposition technique is very apparent, but so are the limitations. All of these are discussed.

5.1.1 Validation

Figure 5.2 shows the comparison of a locally generated pair distribution plot to published results from another molecular dynamic simulation. The baseline, as mentioned, is from Allen and Tildesley [36] and was generated by a leap-frog Verlet algorithm. The two results are very close. The new version of the velocity Verlet, re-ordered to remove the acceleration array, is therefore validated. Also, the code appears free of any boundary defects as these should appear as anomalies as well.

5.1.2 Performance Comparisons

The truncated octahedron code developed for the diffusion simulations is compared here to cubic boundary codes developed by Plimpton [32]. Chapter 4 discussed the various parallel concepts presented by Plimpton, but this same article also contains a benchmark timing of a bulk liquid simulation on a wide array of computer architectures. The liquid structure has a reduced density of $\rho^* = 0.8442$ and a reduced temperature of $T^* = 0.72$. Argon is simulated here with a force cutoff of 2.5σ. A combination linked list and Verlet list with a list radius of 2.8σ is implemented to provide optimal performance on large systems. The time step used is 10 femtoseconds, and the Verlet list is updated at a constant value of 20 steps. An atom decomposition technique is used in the parallel runs, while a vectorized serial algorithm from Grest et al. [47] is used on the single processors of the Cray Y-MP and Cray C90. The parallel runs were performed on the Cray T3D, the nCUBE 2, the Intel iPSC/860, and the Intel Paragon [32].

Figure 5.3 shows the comparison of the performance of the truncated octahedron code on the IBM SP2 to Plimpton's serial runs on the Cray Y-MP and Cray C90. As shown, the parallel code beats the Cray Y-MP performance on just 8 processors, while the stronger Cray C90 performance is surpassed with the 24 processor set. Plimpton states that these performance values of the serial code are the fastest known to date for the molecular dynamics simulations on conventional vector supercomputers. Since the article was updated in June 1994, these comparisons reveal that the SP2 code performs quite well.

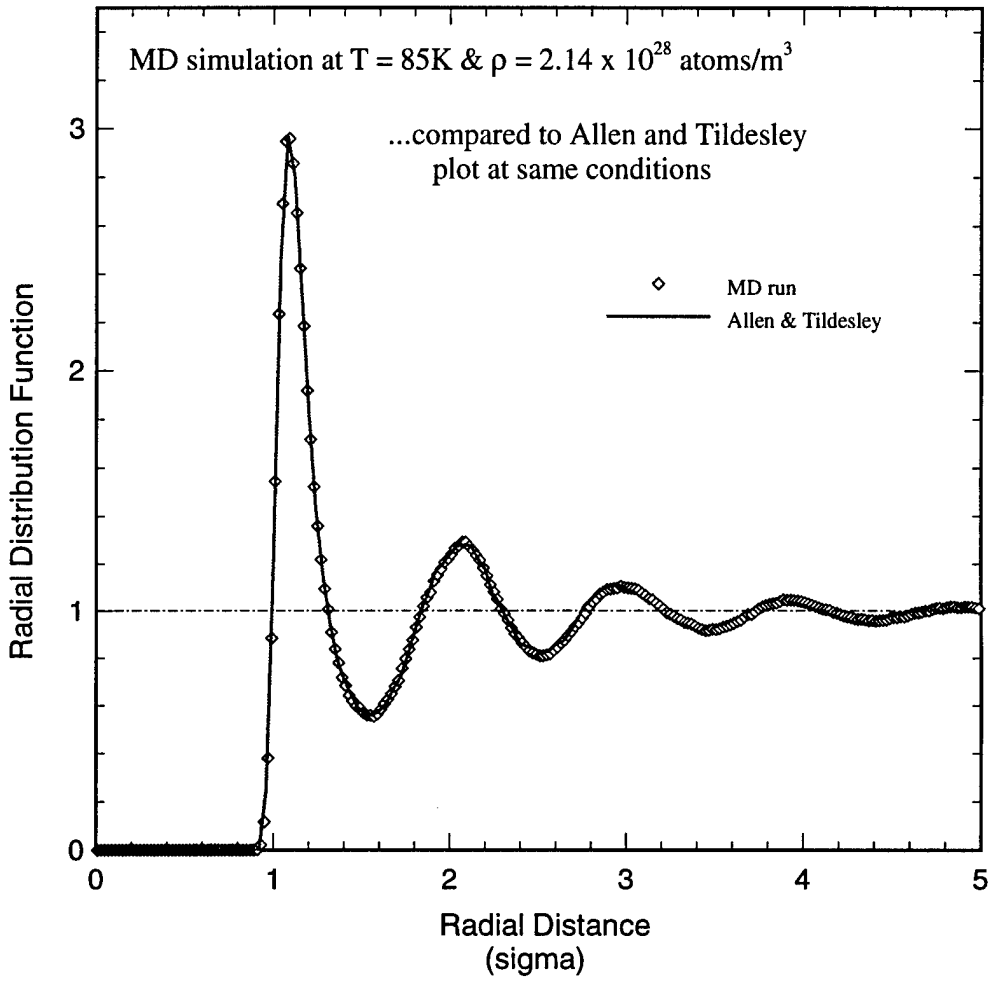


Figure 5.2. Radial distribution function comparison. Present data compared to Allen and Tildesley [36].

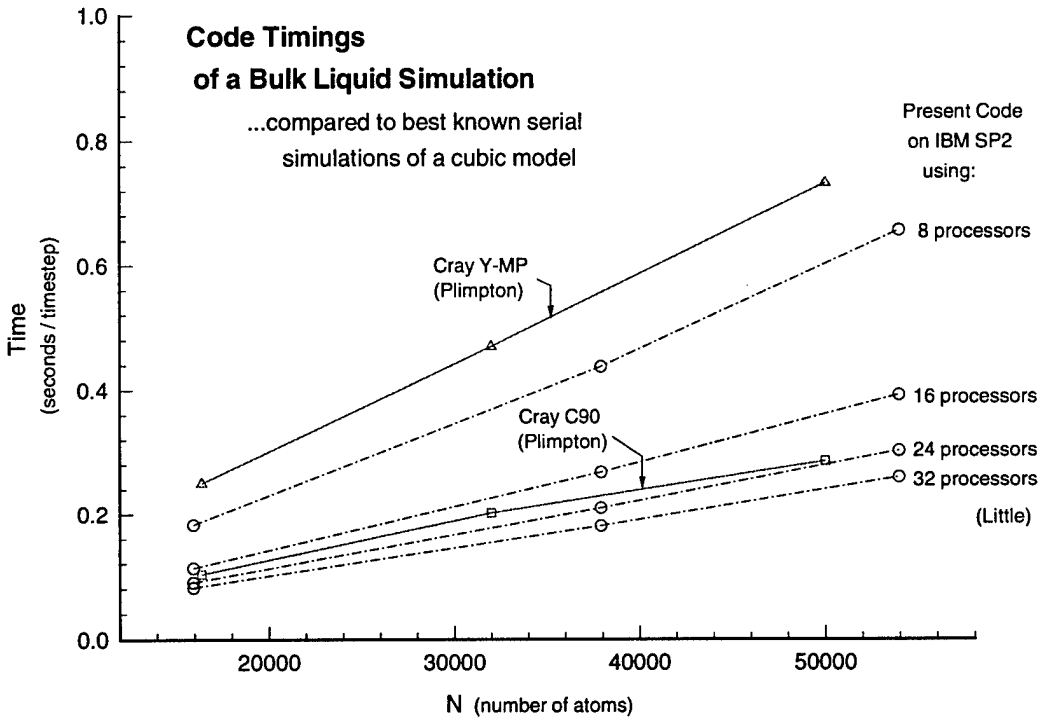


Figure 5.3. Code performance compared to best serial codes.

The benchmark problem was also run on various parallel architectures. The timings for a particle decomposition code are shown in figure 5.4. The present code performance on the 32 processor IBM SP2 is also included in the plot. Again, the truncated octahedron code performs fairly well. The linear scaling with problem size evidenced by the benchmark runs is matched by the SP2 case. Also, the simulation rate is consistently double the rate of the 512 processor nCUBE 2 and nearly four times faster than the 32 processor Intel iPSC/860. Only the 512 processor Intel Paragon and the 256 processor Cray T3D show better performance. However, these machines have peak speeds of

512 processor Intel Paragon:	38 Gflops
256 processor Cray T3D:	38 Gflops
32 processor IBM SP2:	8 Gflops

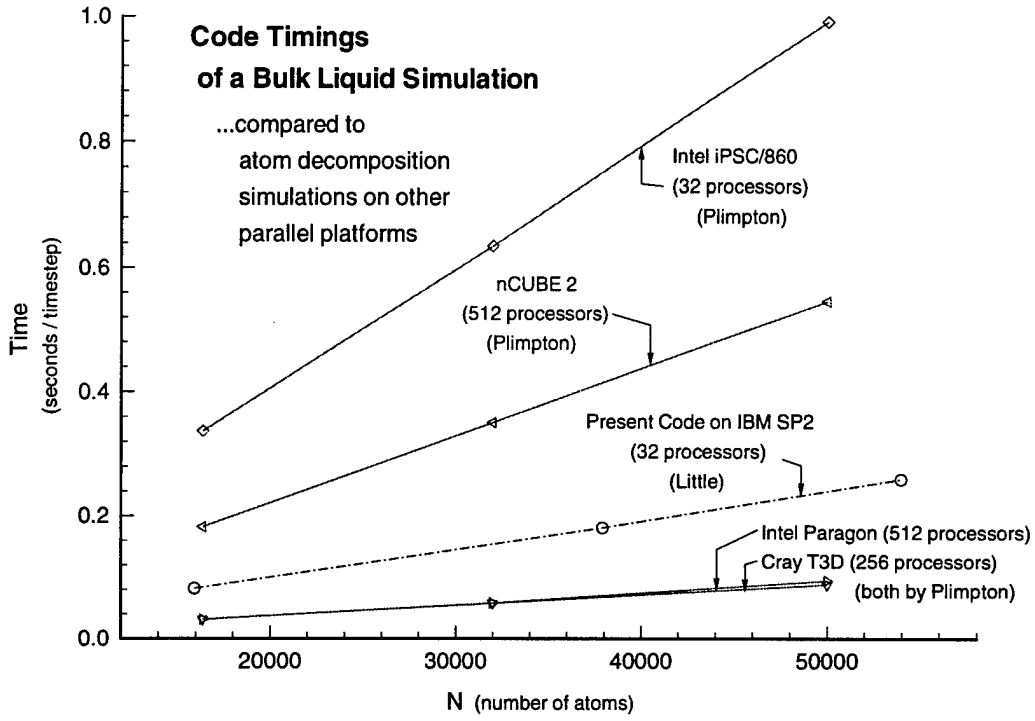


Figure 5.4. Parallel code performance comparisons. Code performance on the IBM SP2 compared to optimized codes on other parallel architectures.

So while the Cray and Intel machines are 5 times faster (theoretically) than the IBM, they only run about 2 times faster.

This poor performance is partially due to the poor communications scaling associated with the atom decomposition technique. Plimpton also shows that a force decomposition algorithm run on these large processor sets can reduce these times by another 25% for the Cray T3D and 40% for the Intel Paragon. Also, doubling the processor sets for the benchmark runs reveals virtually no improvements using the particle decomposition technique, but the more scalable force decomposition code shows promising results. A 512 processor Cray T3D and a 1024 processor Intel Paragon yield additional savings of nearly 40% and 25% respectively. These timings provide incentive to expand the droplet diffusion research onto these more massively parallel

platforms with a force decomposition code. But the strong computational performance on each processor of the SP2 reveals this platform is still very competitive.

As mentioned, all of these benchmark comparisons are for bulk liquid simulations. The diffusion problem consists of distinctly varied density profiles. The droplet requires significantly more computations than the lower density surroundings. The reader is reminded that this load is proportional to the square of the density, so even the relatively compact near-critical environment displays this load imbalance across the geometries. The particle decomposition technique was specifically developed to meet this challenge.

Figure 5.5 shows the success of the technique. The data plotted here is from 500 time steps of the simulation of the large droplet diffusion. This 367,000 atom system is partitioned over 16 processors. The code provides efficient and independent timings of the various workloads associated with the simulation on each processor. As shown, the computations and communications are level, and the waiting periods, indicative of load imbalances during each time step, are almost non-existent. The particle decomposition code therefore provides the load balancing desired for the modeling of the diffusion process.

Chapter 3 of this report details that communications scalability is the price associated with the achieving of this balance. The results from the same problem performed on 32 processors are shown in figure 5.6. The same excellent load balancing is shown, but also the communication loads are nearly identical to the 16 processor performance. In fact, only the parallel computations show significant decreases in simulation times. The other workloads are essentially processor independent and will dominate the simulation time over large processor sets. This is also the reason why the particle decomposition performance on the Cray T3D and the Intel Paragon changed very little when the processor levels were doubled.

One more limitation to scalability should also be understood before completing this performance section. The computations scale fairly well with the even loading across larger processor sets, but this scaling is not perfect. The force computations in the simulation are all based on sampling across the full physical domain on each processor. While the number of displaced atoms handled by each processor is evenly

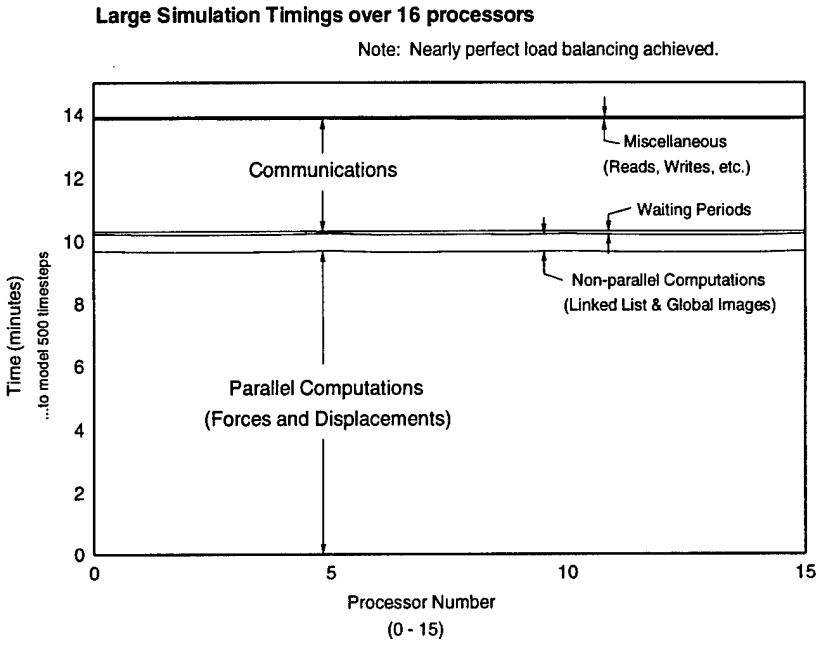


Figure 5.5. Load balancing across 16 processors.

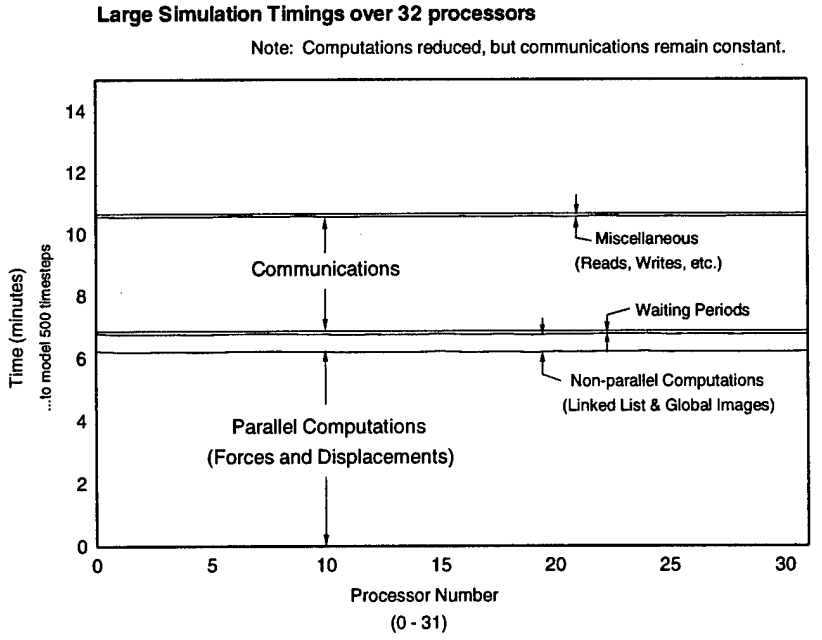


Figure 5.6. Load balancing across 32 processors.

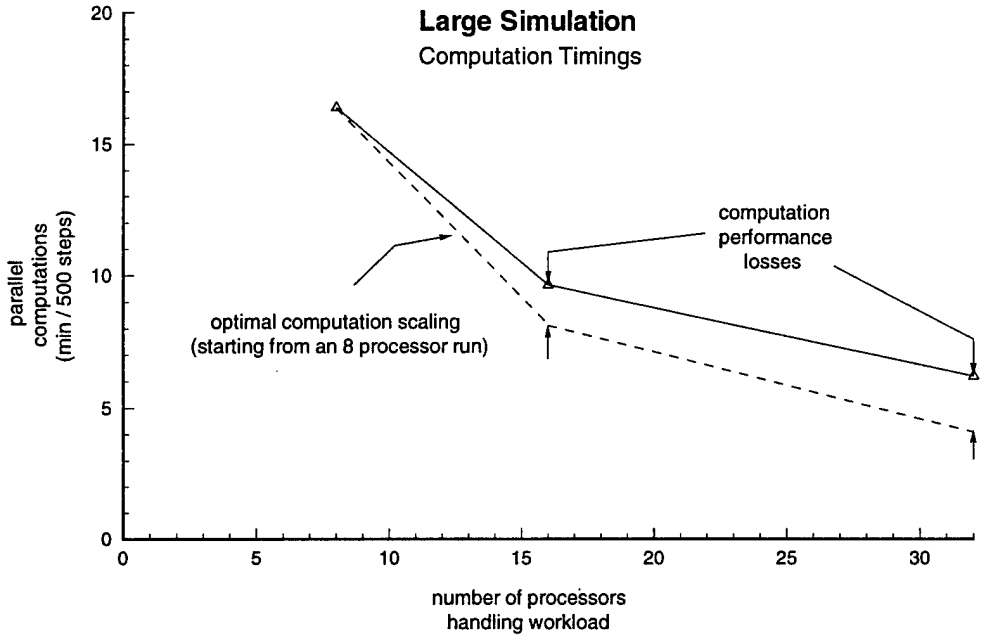


Figure 5.7. Computation times for the large supercritical simulation. The cost of the loop structure is a constant and becomes more important over increasing processor levels.

divided, the loop structure associated with the neighbors is independent of processor levels. As a result, the number of cache misses and other loop structure workloads are additional limiting factors to scalability. This characteristic is displayed for the large simulation run in figure 5.7. As shown, over a minute of the anticipated computational savings for the 500 time steps is consistently required to cover these computation efficiency losses. Since three minutes in savings represents the perfect scaling level from 32 to 64 processors, a one minute penalty certainly becomes significant.

The diffusion code utilized in this thesis therefore sees very limited performance gains beyond 32 processors. The reader should be aware, however, that monatomic simulations result in relatively small computational loads. The future research will involve more detailed computations of increasingly complex molecules. The required loop structures and communication loads will, however, increase only slightly. So

the results detailed herein show a worst case scenario with regards to code scalability. While the force decomposition technique may become warranted, the particle decomposition code will likely remain an important building block for future research.

5.2 Simulation Results

This section details the simulation results corresponding to the diffusion of a 5,600 atom droplet into four different environment conditions. The droplet, equilibrated to saturated conditions at 100 K, is shown on the property chart of figure 5.1 as the data point located on the saturated liquid line. The pressure at this point is 0.32 MPa. The four environment conditions are split into two subcritical and two supercritical cases. Further details associated with these simulations are provided in their respective subsections. Details of the simulation output parameters are provided in the remainder of this introduction of the simulation results.

Subcritical droplet evaporation can be approximated using an approach termed the D^2 *evaporation law* [1]. The D refers to the droplet diameter. The law states that the square of the diameter will diminish linearly during the diffusion process. This occurs because the drop retains a spherical structure throughout the process. The surface area, exposed to the energetic surroundings, is therefore

$$A_s = 4\pi {r_s}^2 = \pi {D_s}^2 \tag{5.1}$$

where r_s is the droplet radius. The thermal and mass concentration gradients are steady and they generate constant fluxes at the droplet surface. Therefore the diffusion of the droplet into the surrounding environment is proportional to the surface area, and a plot of the square of the droplet diameter is a straight line. A similar diffusion rate presentation is desired for the molecular dynamic simulations. This would provide a means of comparing D^2 law rates to computational simulation results.

Measuring the diameter of the droplets during the molecular dynamic diffusion simulations was initially performed by defining a surface using a technique by Maruyama [22]. He measures the local atomic density for each atom using the cutoff sphere as the sampling domain. Any atom whose resultant density value is within a

range of 25% to 75% of the droplet interior density is defined as a surface particle. Since the droplets are periodically centered about the computational origin in the MD simulations, the droplet radius can be measured as simply the average radial position of these surface atoms. This technique proved useful for the subcritical simulations, but was ineffective for the supercritical tracking. During these cases the droplets quickly loose their spherical geometry due to the surface tension reduction. The measuring of an average radial position is then misleading.

An alternative means which both matched the subcritical radius tracking and provided a logical means of following the supercritical progress was developed to counter the problem. Instead of using a measured surface radius, the volume of the liquid structure is tracked as the enumeration of atoms whose local densities exceed a given level. This value, termed N_{drop}, can be regarded as an indicator of droplet volume since the drop is simply the total collection of these particles. Modifying the parameter further to ${N_{drop}}^{2/3}$ provides a similar indicator of surface area.

The utility of this parameter when evaluating subcritical diffusion is easily demonstrated. Assuming the droplet interior is at a relatively uniform density during the diffusion, ${N_{drop}}^{2/3}$ is proportional to D^2 during a subcritical run. Since the drop retains spherical symmetry, D^2 can be approximated using the spherical volume formula as follows.

$$V_{drop} = \frac{4}{3}\pi {r_s}^3 = \frac{1}{6}\pi D^3 \approx \frac{N_{drop}}{\rho_{drop}} \tag{5.2}$$

as

$$D^2 = \left(\frac{6N_{drop}}{\pi\rho_{drop}}\right)^{2/3} = \left(\frac{6}{\pi\rho_{drop}}\right)^{2/3} {N_{drop}}^{2/3} \tag{5.3}$$

Here V_{drop} is an estimate of the droplet volume based on the tabulated N_{drop} and an assumed uniform droplet density, ρ_{drop}. This allows the comparison of the ${N_{drop}}^{2/3}$ values from a series of times in the diffusion simulation to a D^2 law computation of the evaporation rate.

The ${N_{drop}}^{2/3}$ also provides a parameter for tracking the progress of the non-spherical supercritical diffusion. Since the ultimate goal of the simulation is the

modeling of droplet combustion, the determination of the liquid structure based on density rather than geometry is a natural choice. The combustion process will proceed only when there is a nearly stoichiometric mix of the fuel and the oxidizer. For subcritical diffusion, this mixing point occurs at the spherical surface. With the supercritical case, however, the burning would occur at the convoluted surfaces, and ${N_{drop}}^{2/3}$ remains an area factor associated with this shape. It is no longer quantifiable as a fixed geometry, but the parameter does give a means of tracking the diffusion process.

The results from the four simulation runs are presented using a combination of the ${N_{drop}}^{2/3}$ regression factor and the contour plot visualizations introduced in Chapter 3. Density, temperature, and the qualitative surface tension measure are all presented. Also, the pair distribution function of the droplet core is used to provide added insight. Before continuing with the presentation of the results, however, a clarification of the contour imaging is provided.

While three dimensional data is available for the contours, the most informative profiles are found along a central plane in the computational domain. Since the droplets are periodically re-centered in all of the simulations, the central plane images show a symmetric slice of the diffusion process. Figure 5.8 shows the central plane geometry. Since the plotted data is averaged using the linked cell structure, contours extend beyond the truncated octahedron shape. The images outside of this shape are simply represented as zero value fields. Also shown in this figure is the variable focus utilized for the four simulation presentations. The environment dimensions are based on limiting their density increase during the evaporation process to less than 30%. As a result, the lower density simulations comprise much larger simulation volumes. To keep the contour images of the diffusing droplets at the same relative scale, the magnification of the image is adjusted accordingly.

5.2.1 Subcritical Diffusion

The critical point for argon is defined as 151 K and 4.9 MPa [7]. A supercritical environment, therefore, must exceed both of these property values. The simulation of the subcritical evaporation of the 5,600 atom droplet consisted of two different

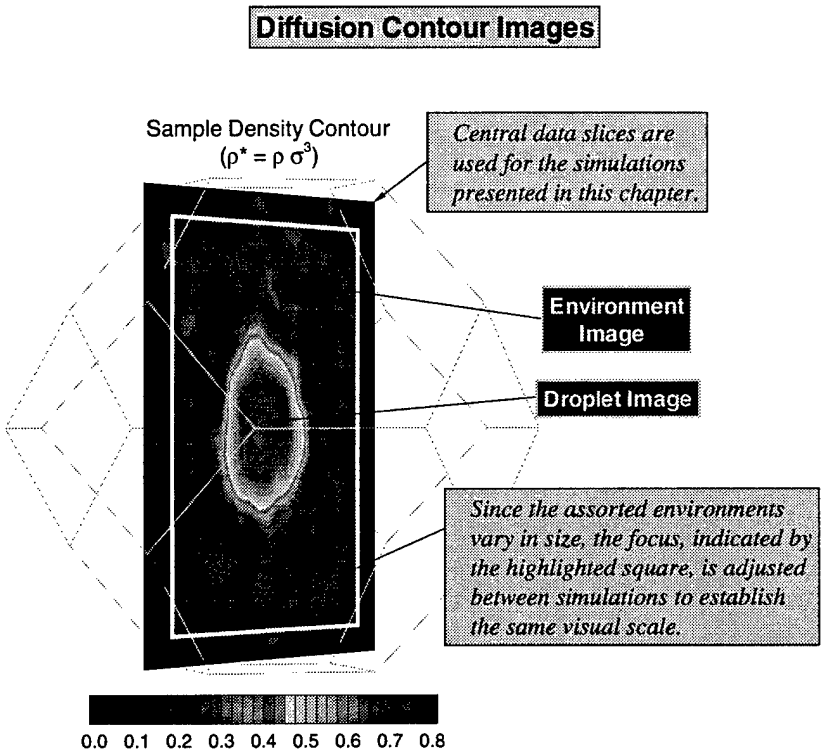


Figure 5.8. Simulation contours. As shown, the contour results for the assorted simulations represent central plane data.

cases. The first, with the surrounding conditions set at 140 K and 3.0 MPa, provides insight into the diffusion with the surroundings below both the critical temperature and pressure. In the second case, however, an environment below the critical pressure, but significantly above the critical temperature is used. The thermal and pressure conditions are 300 K and 3.0 MPa respectively. Both of these simulations provide encouraging levels of insight and simulation validity.

The first subcritical simulation, with the low thermal condition of 140 K, was the slowest diffusion simulation. The $N_{drop}{}^{2/3}$ regression plot (figure 5.9) shows the complete evaporation process requires nearly 2000 psec (and 200,000 time steps). The plot also reveals a straight line regression which qualitatively matches expectations. The small thermal gradients make comparisons to quantitative D^2 theory difficult.

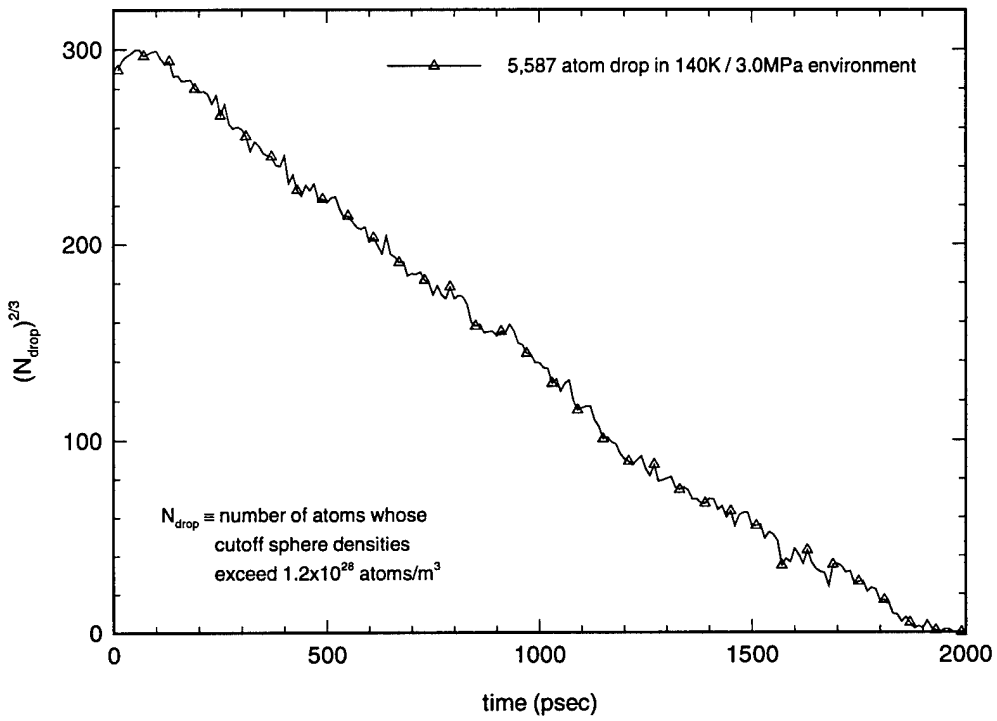


Figure 5.9. Subcritical regression plot.

The surface temperature of the droplet is within a degree of the environment temperature. This is simply too close for any hope of accuracy in the D^2 computations. Additional analysis of this process, however, does reveal further matches with anticipated thermodynamic behavior.

As previously mentioned, the evaporating droplet is initially set to saturated conditions at 100 K and 1200 kg/m^3. The pressure in the drop is therefore assumed to match the saturated level as well at 0.32 MPa. When this drop is exposed suddenly in the simulation to the higher pressure environment at 3.0 MPa, it should adjust accordingly to saturated conditions at this new pressure. Once this adjustment is reached, the droplet properties will remain essentially constant, and the energy from the hot surrounding gas will contribute exclusively to the diffusion process. This droplet property shift is depicted in figure 5.10. The evaporating surface exists as a narrow band of rapidly changing density from the saturated liquid to the saturated

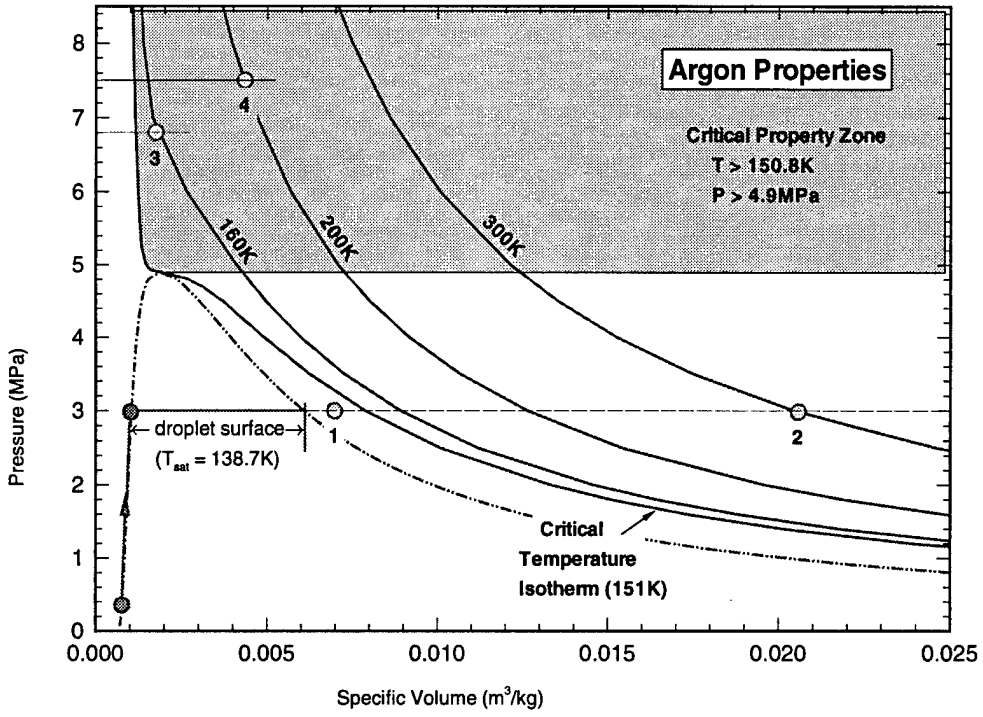


Figure 5.10. Subcritical simulation conditions. The droplet quickly adjusts to the 3.0 MPa environment and the diffusion occurs across the saturation line.

vapor states at 3.0 MPa. The temperature here will correspondingly match the saturated level (138.8 K), while the droplet interior will exist as slightly sub-cooled. The density in the interior, however, will remain very close to the saturated liquid value of 960 kg/m^3.

Figure 5.11 shows the tracking of the core pair distribution profile at three different periods in the simulation. As shown, the core retains a decidedly liquid structure. The slight compression of the second and third plots as compared to the first is due to the increased temperature and reduced density. The progression of the core properties is also shown for these three time periods. These values explain why the second and third distribution profiles match so closely; the core has reached the saturated point in accordance with the 3.0 MPa environment. The sub-cooled interior temperature is here shown to reach a value of 123 K.

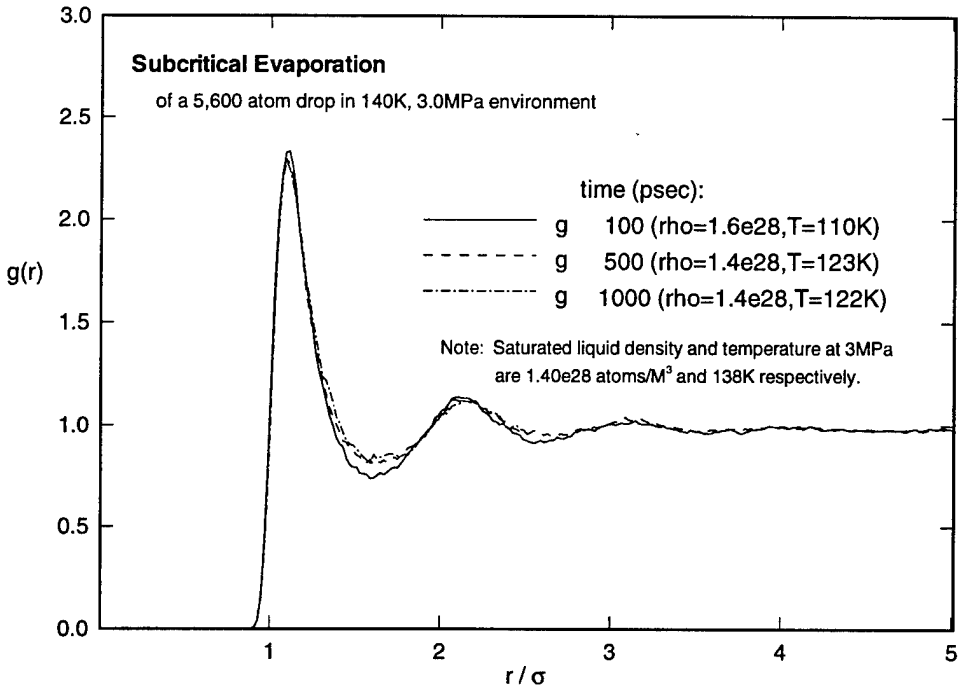


Figure 5.11. Subcritical liquid core tracking. The pair distributions show the anticipated liquid core and the densities match the saturated liquid state at the environment pressure.

The contour profiles for this case are shown in figure 5.12. They show a time series of four image sets during the full evaporation process. This and subsequent contours are presented in color to clearly provide the associated details of these plots. The attainment of the saturated density in the droplet core is indicated by the reduction of the density level from red to a mix of yellow and orange (a reduced density of 0.55 correlates to the saturated liquid condition at 3.0 MPa). The series of thermal images are not very informative. The environment temperature is simply too low to allow statistically viable imaging of the thermal gradients. The surface tension visual is more useful. The initial density of the subcritical environment is 159 kg/m^3. Even though the condition is subcritical, the close proximity to the vapor curve at the elevated pressure (see figure 5.10) results in this relatively high environment density. The contour image shows the surface tension reduces to a low level during the run as

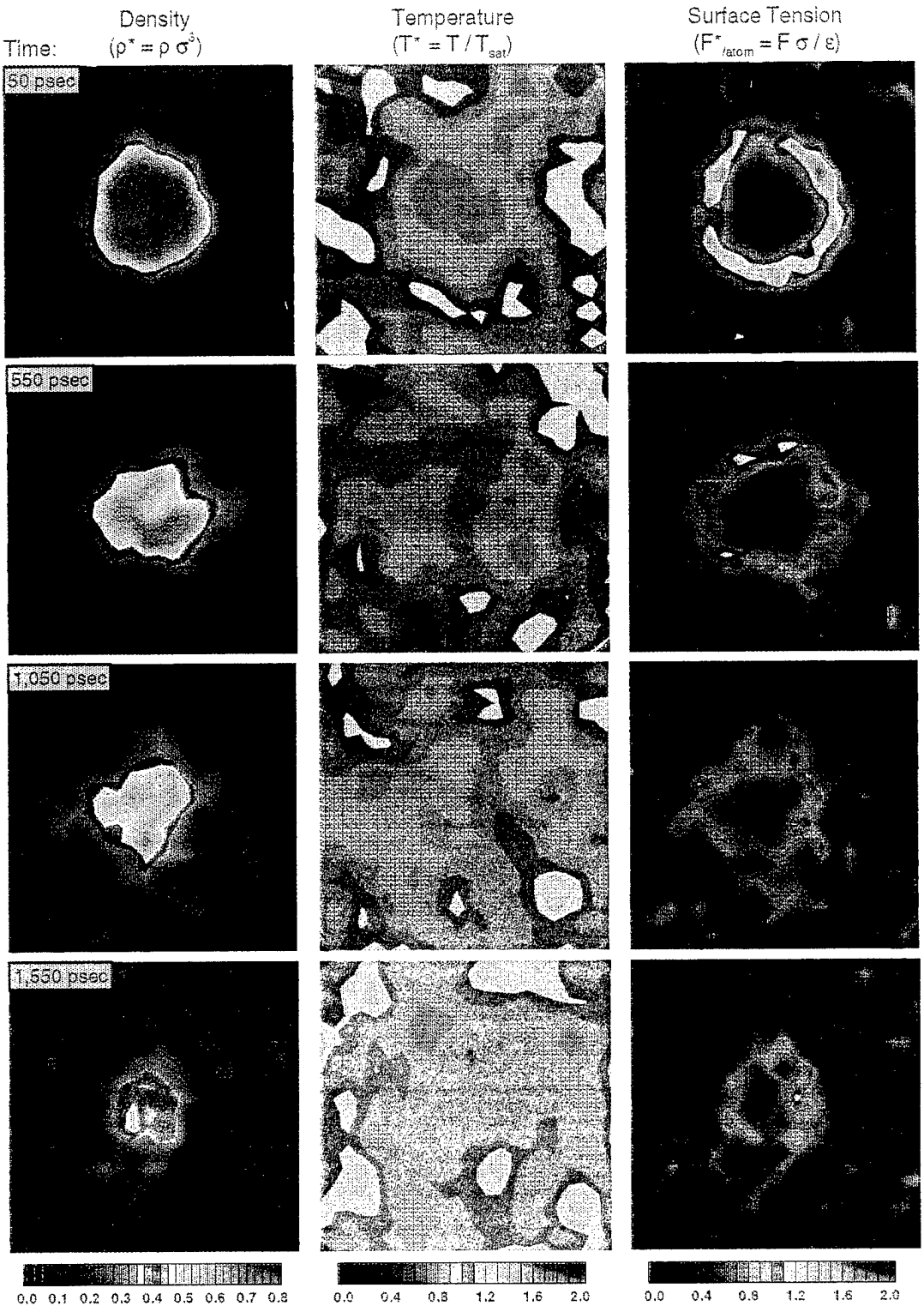


Figure 5.12. Low temperature Subcritical diffusion contours. A 5,587 atom drop equilibrated to 100 K and 1200 kg/m^3 is placed into a 140 K, 3.0 MPa, and 159 kg/m^3 subcritical environment.

a result. The lower surface tension, in turn, causes the droplet to deviate somewhat from the anticipated spherical structure.

The second subcritical environment was specifically chosen to eliminate many of the evaluation problems which were present in the first simulation. The high temperature, yet low pressure, system of 300 K and 3.0 MPa has a density of only 49 kg/m^3. Figure 5.13 shows four sets of contours generated when simulating the 5,600 atom droplet diffusion in these surroundings. The droplet should, once again, shift in response to the 3.0 MPa surroundings. The images reveal both this adjustment and many other interesting results.

The drop structure is clearly imaged in figure 5.13 with spherical symmetry much more readily maintained than in the previous simulation. This is because the low density in the environment allows the surface tension to reach a steady and relatively strong level after the anticipated initial adjustment. The yellow and orange mix attained in the droplet density profiles is once again indicative of the saturated density level at 3.0 MPa. The reader may notice that the color for this case, however, is slightly more intense at the interior than in the previous subcritical simulation. This is because the stronger surface tension causes this interior to exist at a slightly higher pressure level and therefore slightly more dense. This premise is further supported by the fact that the droplet density profiles match the images in the interior of the surface tension rings. This run also shows a thermal layer approaching the saturated temperature surrounds the droplet. The thermal color map is normalized by the saturated temperature of 139 K. The saturated thermal layer is therefore depicted by the thin green contour. The remainder of the droplet is only slightly sub-cooled, again matching expectations. Interestingly, the saturated thermal ring closely matches the surface tension profile and they both exist at the outermost edge of the density image. This indicates the evaporation begins relatively deep into the droplet structure. The vapor temperature is reached only after the surface forces diminish.

The strong thermal and concentration gradients also provide a diffusion profile which strongly matches the theoretical. Figure 5.14 indicates the evaporation is quite rapid compared to the previous subcritical case, but the linearity is even more pronounced. The density cutoff factor in these plots is purposely chosen relatively

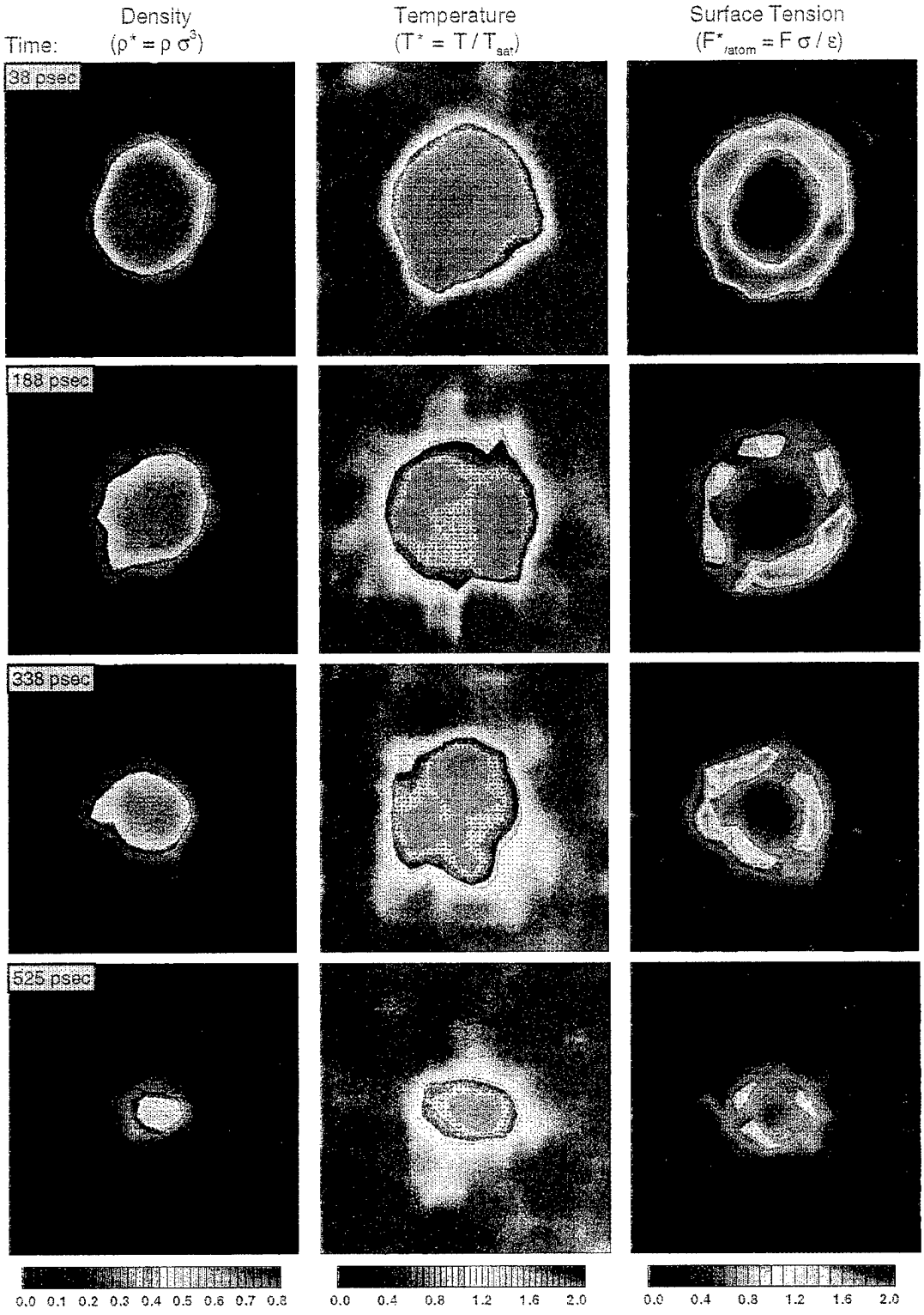


Figure 5.13. High temperature subcritical diffusion contours. A 5,587 atom drop equilibrated to 100 K and 1200 kg/m^3 is placed into a 300 K, 3.0 MPa, and 49 kg/m^3 subcritical environment.

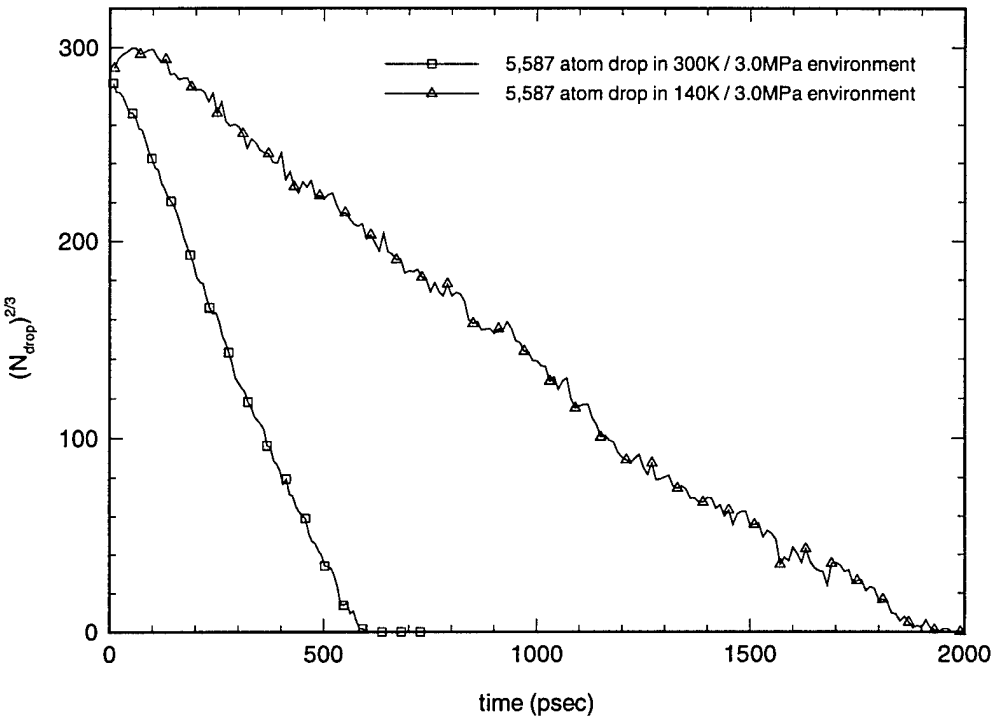


Figure 5.14. Subcritical regression comparisons.

close to the saturated liquid levels at 3.0 MPa. This results in an inclusion of the more uniformally dense inner profile in the N_{drop} enumeration. Since this shape is clearly defined and nearly spherical due to the increased surface tension, the regression plot can be assumed to closely profile the surface area. This then can be compared to a computed regression from the D^2 evaporation law.

For the relatively simple self diffusion problem modeled here, the D^2 regression rate can be computed as [1]

$$\frac{d(D^2)}{dt} = -\beta_v \tag{5.4}$$

where the evaporation coefficient, β_v, is

$$\beta_v = \left(\frac{8\rho_s \mathcal{D}_s}{\rho_{drop}}\right) \ln(1+B) \tag{5.5}$$

and the transfer number, B, can be determined using

$$B = \frac{c_p \left(\mathrm{T}_{env} - \mathrm{T}_s \right)}{h_{fg}} \tag{5.6}$$

In these equations, T_{env} and ρ_{env} are the temperature and density, respectively, of the surrounding environment, in this case 300 K and 49 kg/m^3. Since the product of density and mass diffusivity, $\rho_s \mathcal{D}_s$, can be assumed independent of temperature, the environment based $\rho_{env}\mathcal{D}_{env}$ can be used as a direct replacement. The self diffusivity, $\mathcal{D}_{env}$, is interpolated from experimental data at atmospheric pressure from Hirschfelder et al.[48] and extrapolated to the high pressure condition using a technique suggested by Bird et al. [49]. The resultant value for argon at 300 K and 3.0 MPa is 0.6×10^{-7}m^2/sec. The specific heat, c_p, of argon at these conditions was interpolated from data presented by Lide [50] as 0.56 kJ/kg K. T_s represents the surface temperature which, as previously shown, exists at the saturated condition of 139 K. The latent heat of vaporization, h_{fg}, at the 3.0 MPa condition is provided by Vasserman et al. [2] as 90.0 kJ/kg. Collecting all of this data and applying to equations 5.4 through 5.6 yields

$$\frac{d(D^2)}{dt} = -1.8 \times 10^{-7}\mathrm{m}^2/sec \tag{5.7}$$

As previously mentioned, equation 5.3 can be used to determine a simulation based evaporation rate. Applying the temporal derivative to this equation gives

$$\frac{d(D^2)}{dt} = \left(\frac{6}{\pi \rho_{drop}} \right)^{2/3} \frac{d({N_{drop}}^{2/3})}{dt} \tag{5.8}$$

The density of the drop is assumed constant at a core observed average level of 1.5×10^{28} atoms/m^2. This is slightly higher than the saturated level at 3.0 MPa due to the increased pressure from the surface tension. The rate on the right side of equation 5.8 is simply the slope of the ${N_{drop}}^{2/3}$ regression. Inspection of figure 5.14 yields $-.55$ atoms$^{2/3}$/psec for this value. The simulation based evaporation rate is therefore

$$\frac{d(D^2)}{dt} = -1.4 \times 10^{-7} \mathrm{m}^2/sec \tag{5.9}$$

This represents a 22% slower rate than the prediction from the D^2 evaporation law. Considering all of the assumptions associated with each computation, however, this is considered a close match.

So the subcritical cases successfully provide both perceptive information and simulation validations. Subcritical evaporation, however, is a process already successfully modeled with simpler techniques. The next section details the results from the supercritical simulations. These analytical observations provide perspectives unavailable prior to this study.

5.2.2 Supercritical Diffusion

In a manner similar to the subcritical studies, the supercritical simulations consist of two simulations. The first, the diffusion of the 5,600 atom droplet into a near-critical environment, reveals initial insights into the supercritical process. As a diffusion study, however, the simulation is inadequate. An explanation will be detailed shortly. The second simulation consists, once again, of immersing the 5,600 atom droplet into a supercritical environment, but during this run the 200 K and 7.5 MPa surroundings provide sufficient gradients to reveal a very effective and interesting diffusion process.

The near-critical environment was initialized by setting the system at the critical density and just above the critical temperature. This resulted in a condition at 533 kg/m^3, 160 K and 6.8 MPa. The associated ratios to critical properties are 1.00, 1.06 and 1.4 respectively. When exposed to these conditions, the droplet still must adjust accordingly. In the subcritical case, the diffusion surface reaches a saturated state correlating with the environment pressure. At supercritical pressures, however, there is no saturation state at which the evaporating surface can reside. Mass and thermal continuity principles still must be met, however. The visualizations of the simulation provide insight to help clarify this issue.

A series of contours generated by the simulation are presented in figures 5.15 and 5.16. The first series shows image sets at 10, 30, 50 and 70 picoseconds. At

these very early points, nearly immediate removal of the surface energy is readily apparent. Also, the droplet quickly reaches a thermal condition just under critical. Due to the lack of surface energy, the geometry of the liquid phase rapidly digresses from the initial spherical structure. An apparent diffusion surface, however, at the critical temperature surrounds the irregular density profile. This is demonstrated further in the second series of images in figure 5.16, based on data at 10, 100, 200 and 600 picoseconds. The diffusion process is overshadowed, however, by a cloud-like dispersion. This is the reason why this simulation does not provide a useful basis for diffusion tracking. In fact, attempts at regression plots reveal no discernable evaporation rate.

Before moving on to the second supercritical investigation, one more observation should be addressed. This is associated with the previously mentioned state discontinuity question. Dispite the nearly uniform thermal images and the completely non-spherical density profiles, the liquid region appears to retain a relatively high density level. Figure 5.17 shows where this author believes the state points of the droplet exist during the process. When the droplet initially adjusts to the high pressure and high temperature conditions, it moves up the saturated liquid line. Since a saturation point is not reached, the structure continues to adjust to the high pressure by rising up just to the left of the critical isotherm. Once the pressure equilibrates, the diffusion occurs between this subcooled state and the supercritical environment. The critical isotherm shifts slightly in the direction of lower specific volume (and therefore higher density) with increasing pressure, so the liquid density rests fairly close to the saturated level at 3.0 MPa observed previously. The figure also displays the close proximity based on both the thermal and density levels of the adjusted droplet to the environment. These low gradients cause the diffusion to be overshadowed by the cloud-like mixing.

A similar adjustment of the droplet interior can be observed in the second supercritical simulation. This time the liquid must adjust to a slightly higher pressure of 7.5 MPa, so the resultant subcooled structure is correspondingly more dense. The difference is very slight, however, since the critical isotherm is nearly vertical at this point. The temperature of the new supercritical surroundings is now set to 200 K

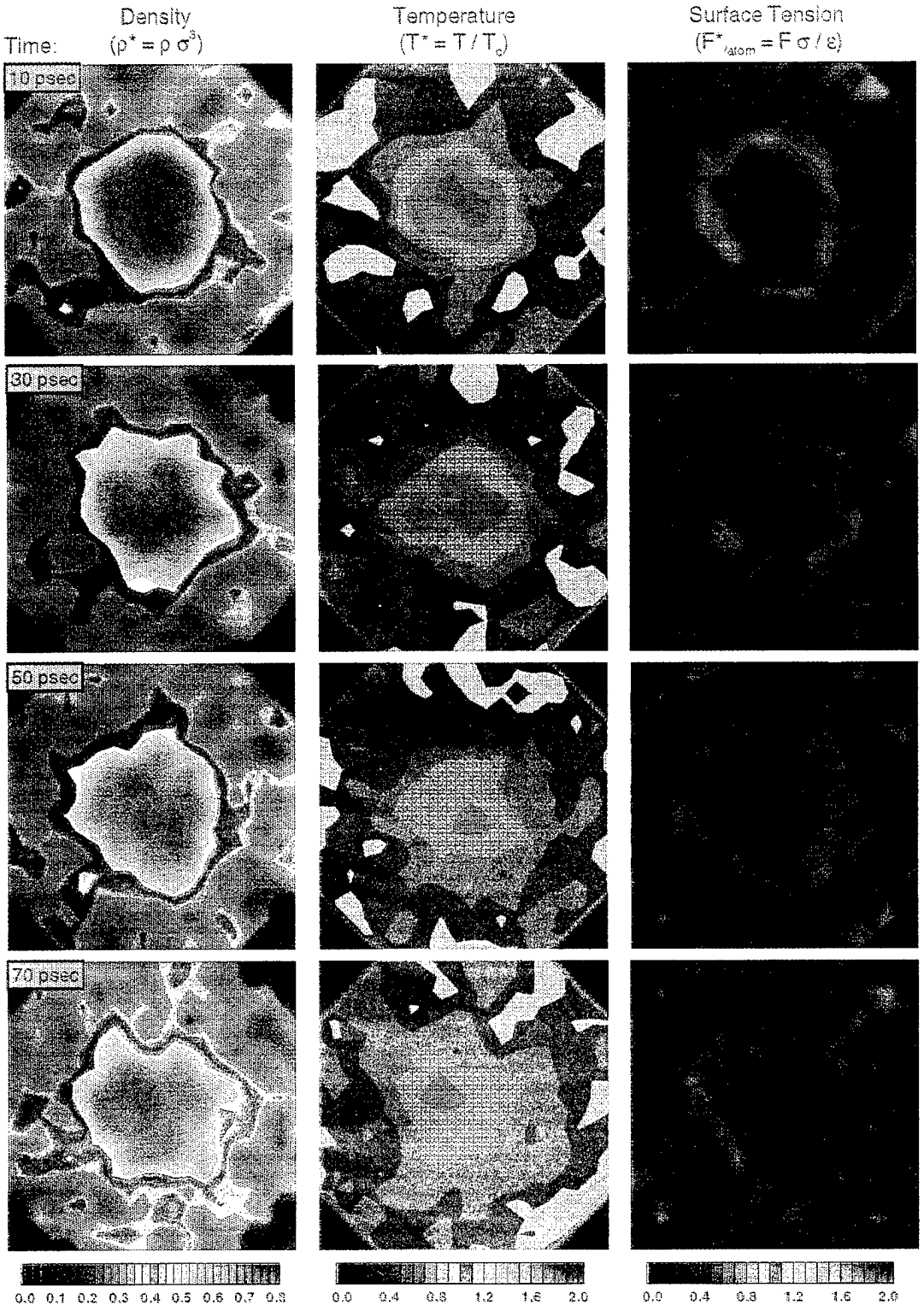


Figure 5.15. Near-critical contours: Initial images. A 5,587 atom drop equilibrated to 100 K and 1200 kg/m^3 is placed into a 160 K, 6.8 MPa, and 530 kg/m^3 near-critical environment.

Figure 5.16. Near-critical contours: Full evaporation. A 5,587 atom drop equilibrated to 100 K and 1200 kg/m^3 is placed into a 160 K, 6.8 MPa, and 530 kg/m^3 near-critical environment.

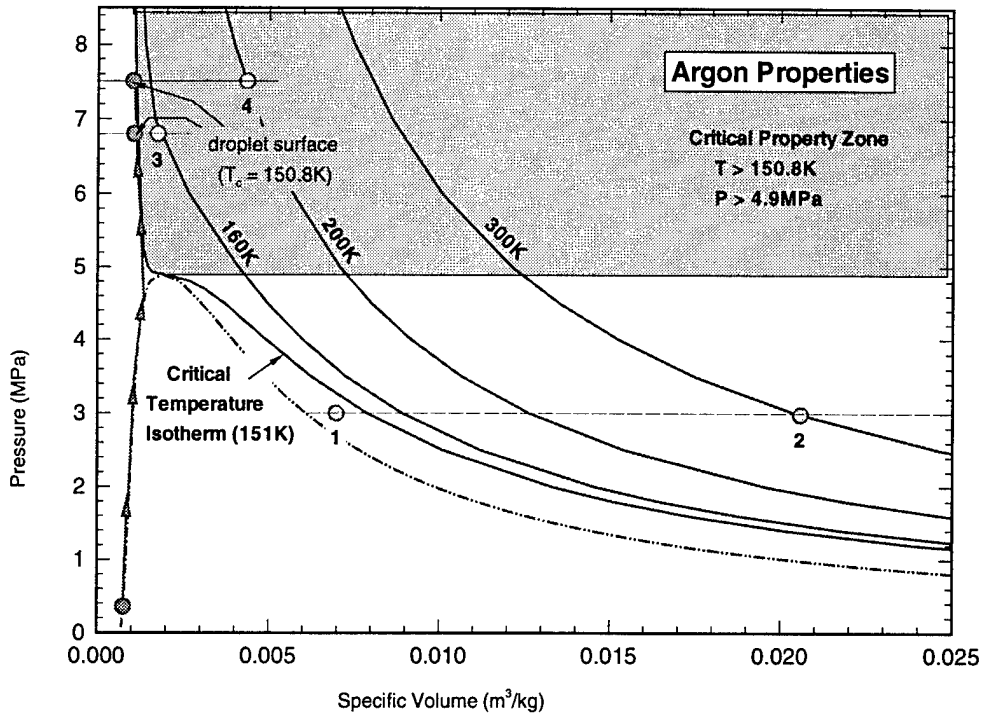


Figure 5.17. Supercritical simulation conditions. The droplet adjusts close to the saturated liquid profile and then just to the left of the critical temperature isotherm when exposed to the supercritical temperatures and pressures of the third and fourth simulation environments.

with a resultant density of 236 kg/m^3. Due to this lower environment density and higher temperature, the thermal and concentration gradients are now stronger. As a result, the supercritical diffusion process can be observed.

A double presentation of contour series, similar to those shown in the previous supercritical discussion, are provided in figures 5.18 and 5.19. The short time span series is now set at 10, 40, 70 and 100 picoseconds. The surface energy still dissipates rather quickly and the spherical geometry breaks down as a result. This effect is not as strong as in the near-critical simulation, however, since the environment is not as dense. The cloud-like dissipation is correspondigly reduced. Instead, a clear diffusion boundary at the critical temperature is apparent. This surrounds the density profile in a manner very similar to the high temperature, subcritical run. Figure 5.19, with

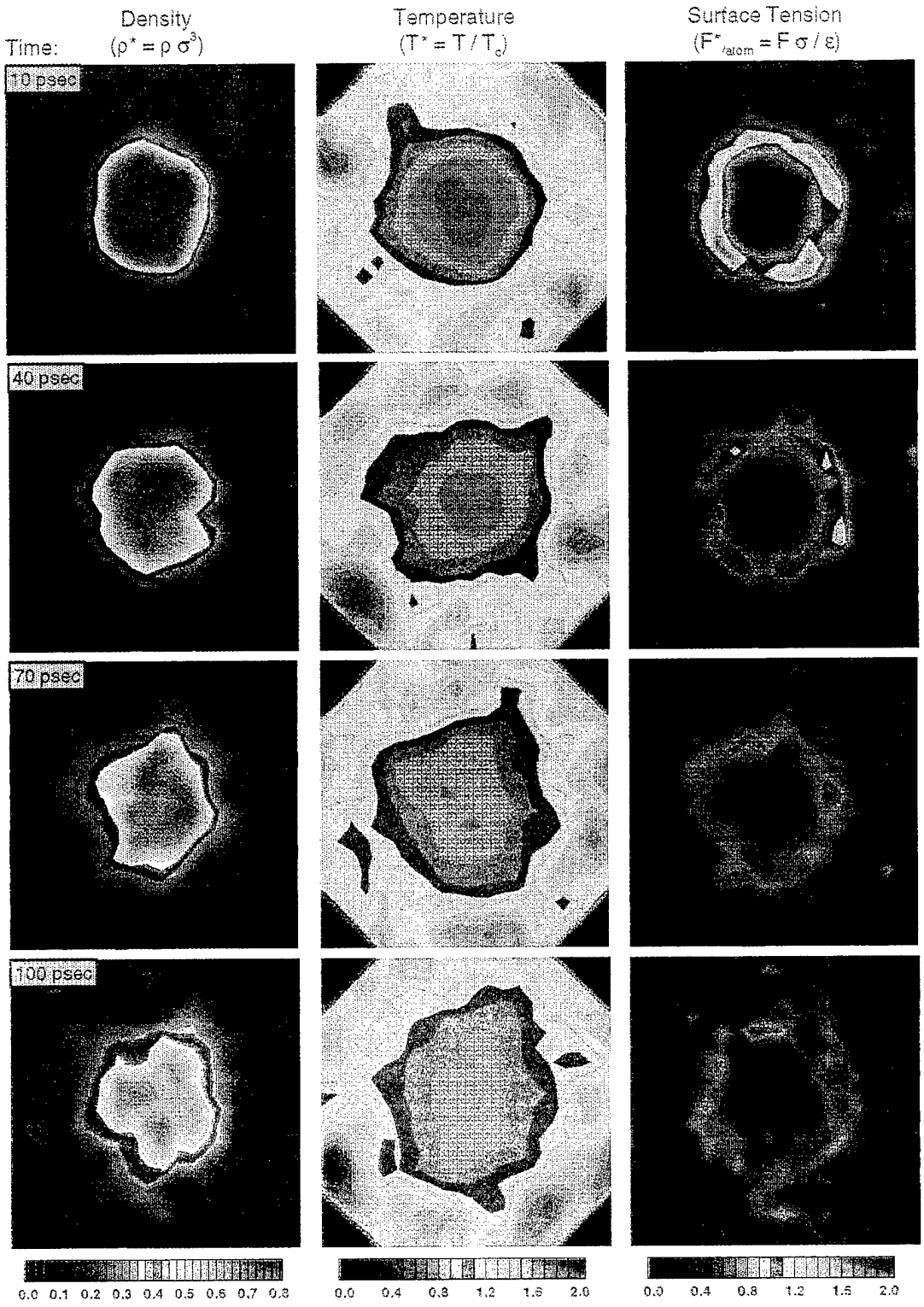


Figure 5.18. Supercritical contours: Initial images. A 5,587 atom drop equilibrated to 100 K and 1200 kg/m^3 is placed into a 200 K, 7.5 MPa, and 236 kg/m^3 supercritical environment.

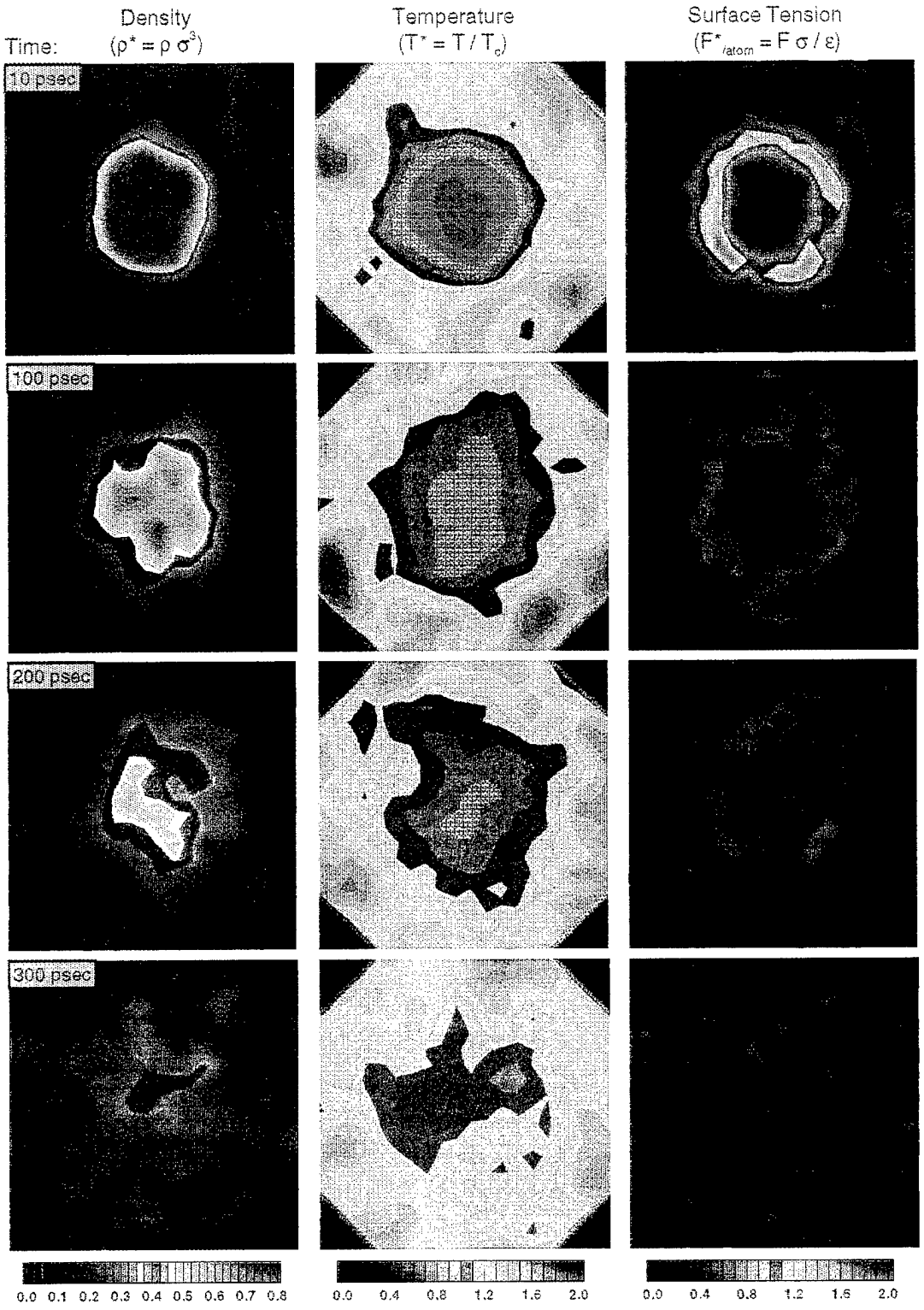


Figure 5.19. Supercritical contours: Full evaporation. A 5,587 atom drop equilibrated to 100 K and 1200 kg/m^3 is placed into a 200 K, 7.5 MPa, and 236 kg/m^3 supercritical environment.

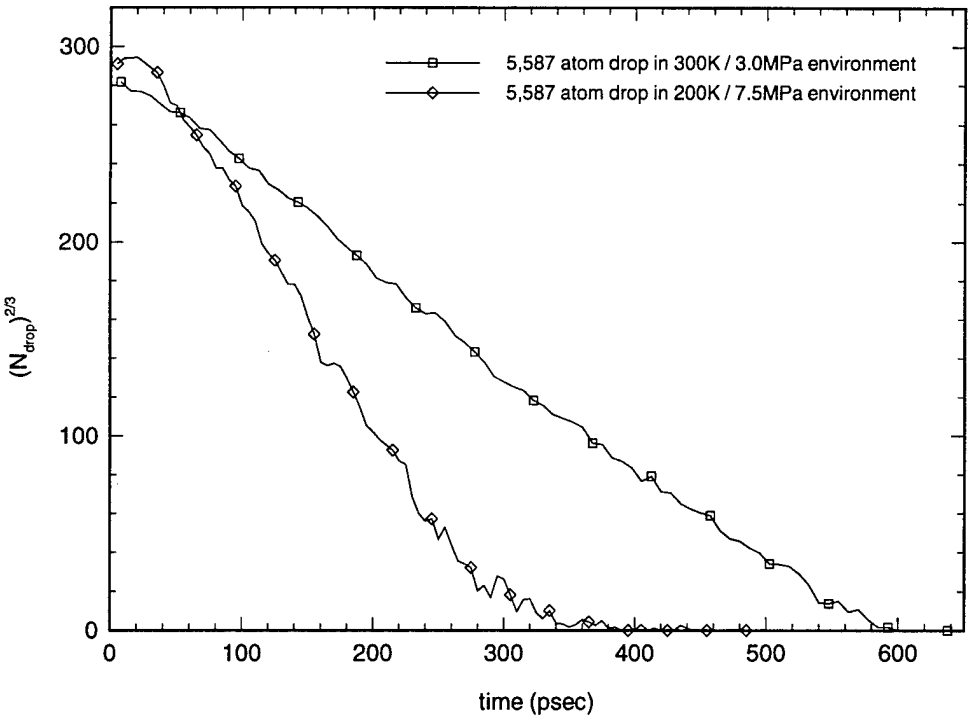


Figure 5.20. Supercritical and subcritical regressions. Comparison of the $N^{\frac{2}{3}}$ regression of the 5,600 atom droplet in the 200 K, 7.5 MPa supercritical environment to the regression in the 300 K, 3.0 MPa subcritical surroundings.

images from 10, 100, 200 and 300 picoseconds, reveals that while this diffusion contour becomes rather irregular, it continues surrounding a subcooled structure throughout the process.

The non-spherical liquid structure, as just mentioned, is a result of the surface tension reduction. In the near-critical simulation significant mixing occurs as a result. In this second simulation, an additional benefit is derived. Figure 5.20 shows a comparison of the regression profile of this 200 K supercritical diffusion to that of the 300 K subcritical run. The scale is expanded to more clearly compare these two rapid processes. Interestingly, the supercritical case also achieves a linear profile after a short adjustment. This indicates that although the shape is irregular, an apparent

steady surface diffusion is progressing. In fact this diffusion into the 200 K environment occurs more rapidly than the subcritical evaporation into the higher temperature surroundings. One explanation is that although the atoms in the subcritical case are hitting the droplet with higher speeds, the frequency is less corresponding to the lower density. One more enhancement, however, is present here, the lower surface energy. In other words, there is a much lower net attractive force pulling the surface atoms towards the droplet structure. Not only are the surface collisions in the supercritical case occuring more often, they are also more likely to discharge surface elements as a result. These results were the genesis of the surface tension description provided in Chapter 3 which detailed the benefits of supercritical environments.

5.3 Scaling Studies

The previous results were very informative, but the 5,587 atom droplet, the evaporating subject in all four simulations, is far from a representative drop in a combustion chamber. With a diameter on the order of just 9 nanometers, this drop is microscopic. Investigations of its evaporation would seemingly be of little use for any practical engineering work. To address this significant problem, two additional simulations were performed. Both were set at the same conditions as the supercritical simulation presented in the last section. The drops were equilibrated to 100 K and then fused into supercritical environments at 200 K and 7.5 MPa. The difference in these new runs is size. The first uses an initial drop of 27,109 atoms while the second uses a 100,570 atom drop. The associated diameters are approximately 16 and 23 nanometers respectively. The results from these larger simulations are important. If a scaling parameter reveals the smaller runs can approximate larger simulations, then the results can be of practical use after all.

Scaling is not a new concept in molecular dynamic simulations. Many references mention the application of small scale studies for macroscopic work. Haile [35] defines the thermodynamic limit using

$$\lim_{\substack{N\to large \\ V\to large}} (\text{simulation results})_{N/V\,fixed} \approx (\text{macroscopic results}) \qquad (5.10)$$

As long as the system density remains fixed, the simulation results can be used to approximate the macroscopic system. This is actually an oversimplification and assumes the microscopic simulation is set at the same thermodynamic state as the desired macroscopic system. With the density set to equivalent levels by fixing N/V, the state can be defined by ensuring similar energy levels.

In the simulations performed for this thesis the system densities are certainly not uniform. Also, the energy levels in the simulations vary significantly due to both the density and thermal profiles. So scaling cannot be performed by simply extending a bulk system concept. Instead, a local density must be considered, and the relative properties must be set appropriately. In other words, if the system is split into subsystems each of which consist of essentially uniform densities and energy levels, then these small pieces can be scaled as noted above. So two simulations, mapping the same relative density and temperature (or kinetic energy) profiles, can be considered scaled representations of identical systems.

The two additional simulations are established based on this scaling principle. Each environment is sized by applying the relative dimensions of the droplet profiles. The droplets are equilibrated in identical fashions and therefore represent scaled density profiles. Also, by setting the same initial temperatures in the environment and the drops, the simulations begin with scaled thermal profiles as required. One point could be argued here, however; the equilibrated drops have the same surface depths. Figure 5.21 shows cross sectional images of the three scaled drops. As seen, the depth of the density gradients at the surfaces are equal, not scaled. This view, however, highlights an important concept in this specific problem.

The results from the 5,600 atom run show that the diffusion process appears to remain a surface effect. So the proper scaling of this simulation should ensure that the surfaces are matched. In other words, rather than ensuring the entire system is an ensemble of small volumes, the elements from the droplet surface are set as the collection. Figure 5.21 depicts this concept by noting that scaled drops should be considered collections of equivalent surface elements. The larger systems simply contain more elements, and macroscopic systems contain very large sets. This concept can be presented in a fashion similar to Haile's representation of bulk scaling as

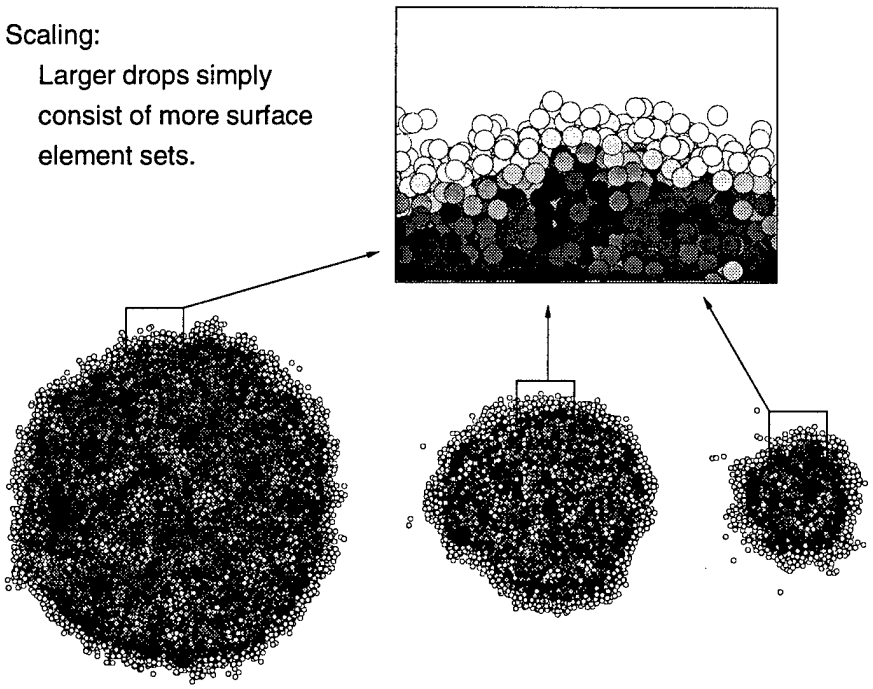


Figure 5.21. Scaling premise.

$$\lim_{\substack{N_s \to large \\ V_s \to large}} (\text{simulation results})_{N_s/V_s fixed} \approx (\text{macroscopic results}) \qquad (5.11)$$

where N_s and V_s refer to the number of atoms and the volume respectively of the droplet surface. Once again, this representation assumes an equivalent energy profile for this region. Additionally, for the droplet diffusion simulation, the environment and interior droplet volumes should be set using the bulk concept to ensure scaled sources exist to drive the surface simulation.

The two larger simulations meet these scaling requirements and should therefore match the supercritical run previously presented. The resultant regression plots are compared to the original smaller case in figure 5.22. The two additional runs generate very encouraging results. The profiles are nearly identical in shape; they appear as

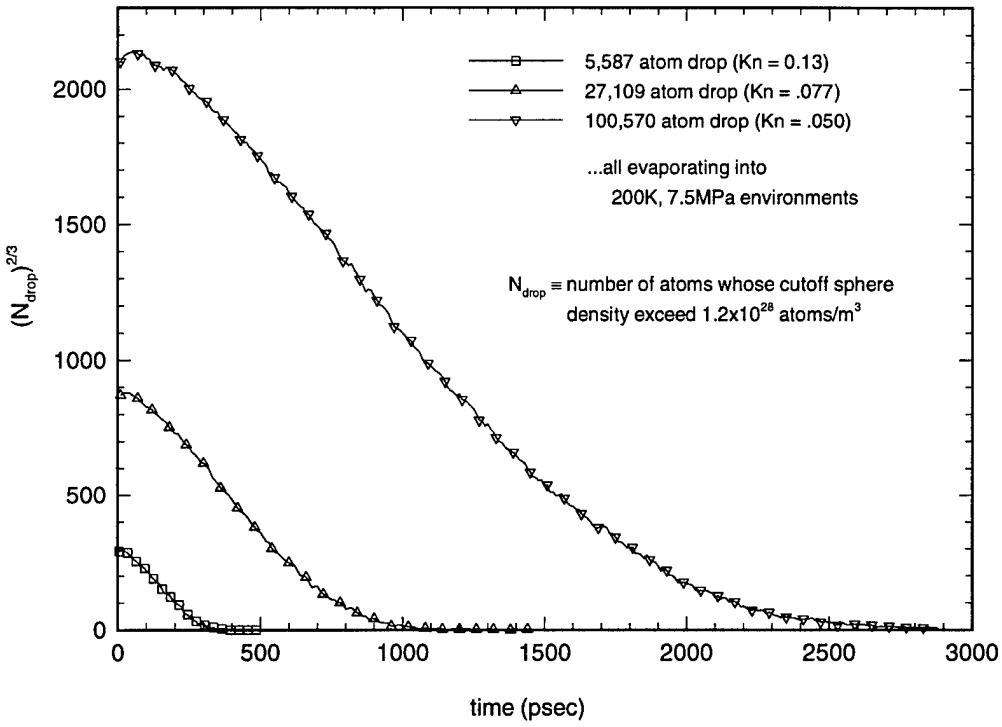


Figure 5.22. $N^{\frac{2}{3}}$ regression plots for increasingly larger simulations.

simply larger representations of the same simulated system. The problem which remains is to determine the scale factor which allows the extrapolation of smaller runs to larger simulations. The contour images give additional information to help meet this challenge.

Early contour profiles from these runs are displayed in figures 5.23 and 5.24. The first is a series of images at 10, 40, 70 and 100 picoseconds for the evaporation of the 27,109 atom drop into the 64,235 atoms comprising the supercritical environment initially set at 200 K and 7.5 MPa. The second set of images are selected from the same times for the evaporation of the 100,570 atom drop into the 266,278 atom environment. Careful comparison of these figures to figure 5.18 reveal that the surfaces do indeed seem to simulate similar effects. Most notably, the surface tension dissipates at the same rate. This results in the irregularities of the density profiles appear to be generated at the same rate as well. Finally, the thermal profile cooling is progressing

at the same depth for each time period. The reader should note that these similarities are not relative rates and depths; they are absolute measures. They do support the scaling concept, however. By envisioning unwrapping the surface images, duplicating according to the relative scaling of the perimeters, and then comparing, one can see the very close similarities.

This visualizing concept also helps postulate the desired scaling factor. Since the surfaces appear to be progressing at very close simulation rates, a measure of these surfaces should provide the answer. In other words, the rate of change of the surface areas, A_s, is a constant for all three simulations.

$$\frac{d\,A_s}{d\,t} = constant \tag{5.12}$$

This equation reveals that the times in the regression plots should be scaled identically as the surface area to retain the constant rate value. Since the ${N_{drop}}^{2/3}$ factor is used as an indicator of this area, it should correspondingly provide the scale factor as follows

$$\text{scale} \equiv \frac{{N_{drop_{s1}}}^{2/3}}{{N_{drop_{s2}}}^{2/3}} \tag{5.13}$$

where the subscripts $s1$ and $s2$ refer to any two scaled simulations. This scale can now be used as a multiplying factor for both the regression and the time variables for the $s2$ regression plot.

The application of this concept is depicted in figure 5.25. By assuming the initial values of the ${N_{drop}}^{2/3}$ parameters are appropriate for the determination of the scale factor (from equation 5.13), the regression plots were scaled up to the large simulation case. The results of this attempt are very apparent in the figure. The two smaller runs accurately depict the large run progression. By simply using the ratio of the initial ${N_{drop}}^{2/3}$ parameters, the entire profile scales quite well.

The contour images of the full diffusions in these larger runs are shown in figures 5.26 and 5.27. The time series are selected to scale appropriately with those displayed in the smallest case in figure 5.19. They verify the close profiling of the three runs using the scaling concept. These images also tend to explain how the volume of the

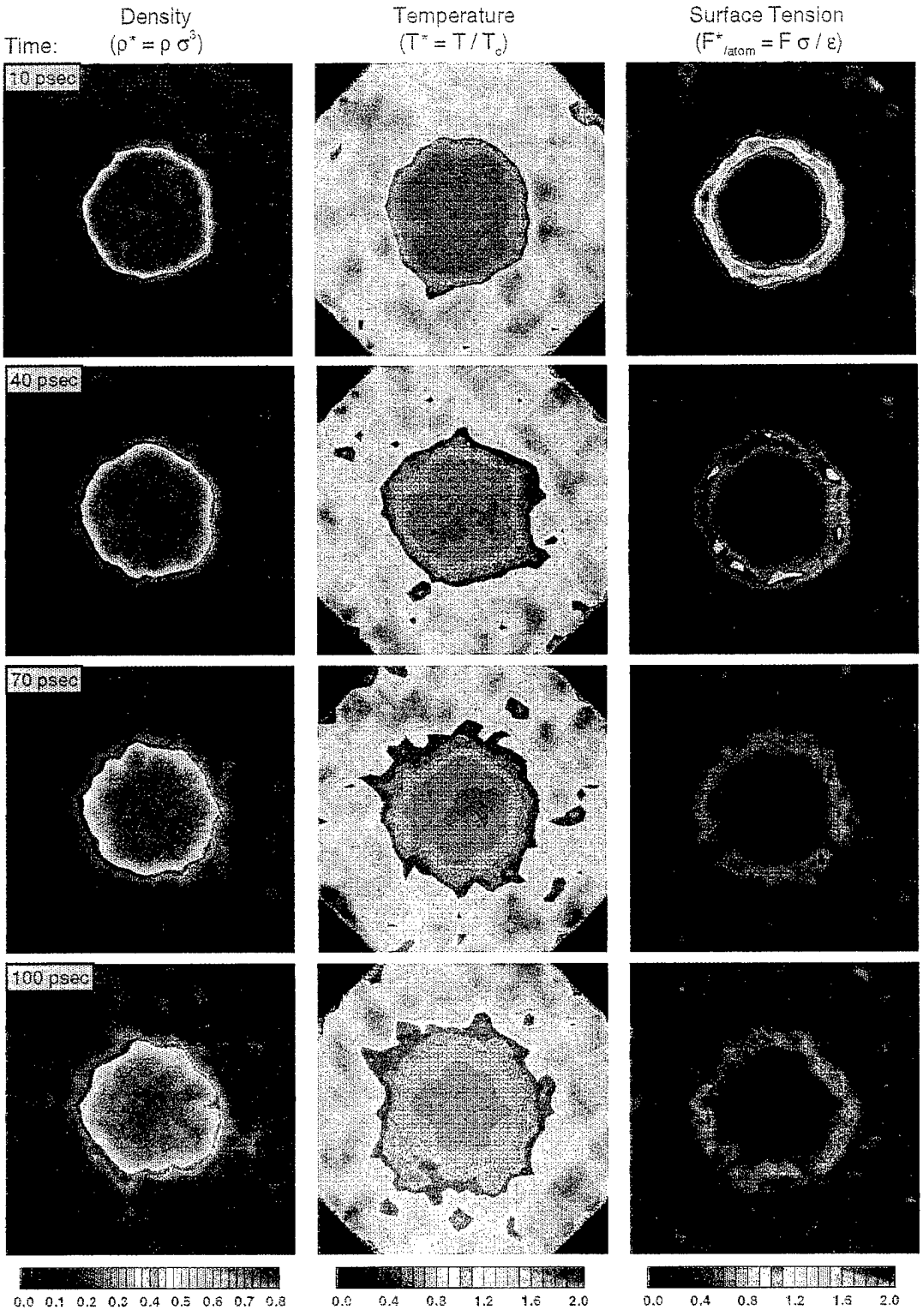


Figure 5.23. Scaling level 1: Initial images. A 27,109 atom drop equilibrated to 100 K and 1200 kg/m^3 is placed into a 200 K, 7.5 MPa, and 236 kg/m^3 supercritical environment.

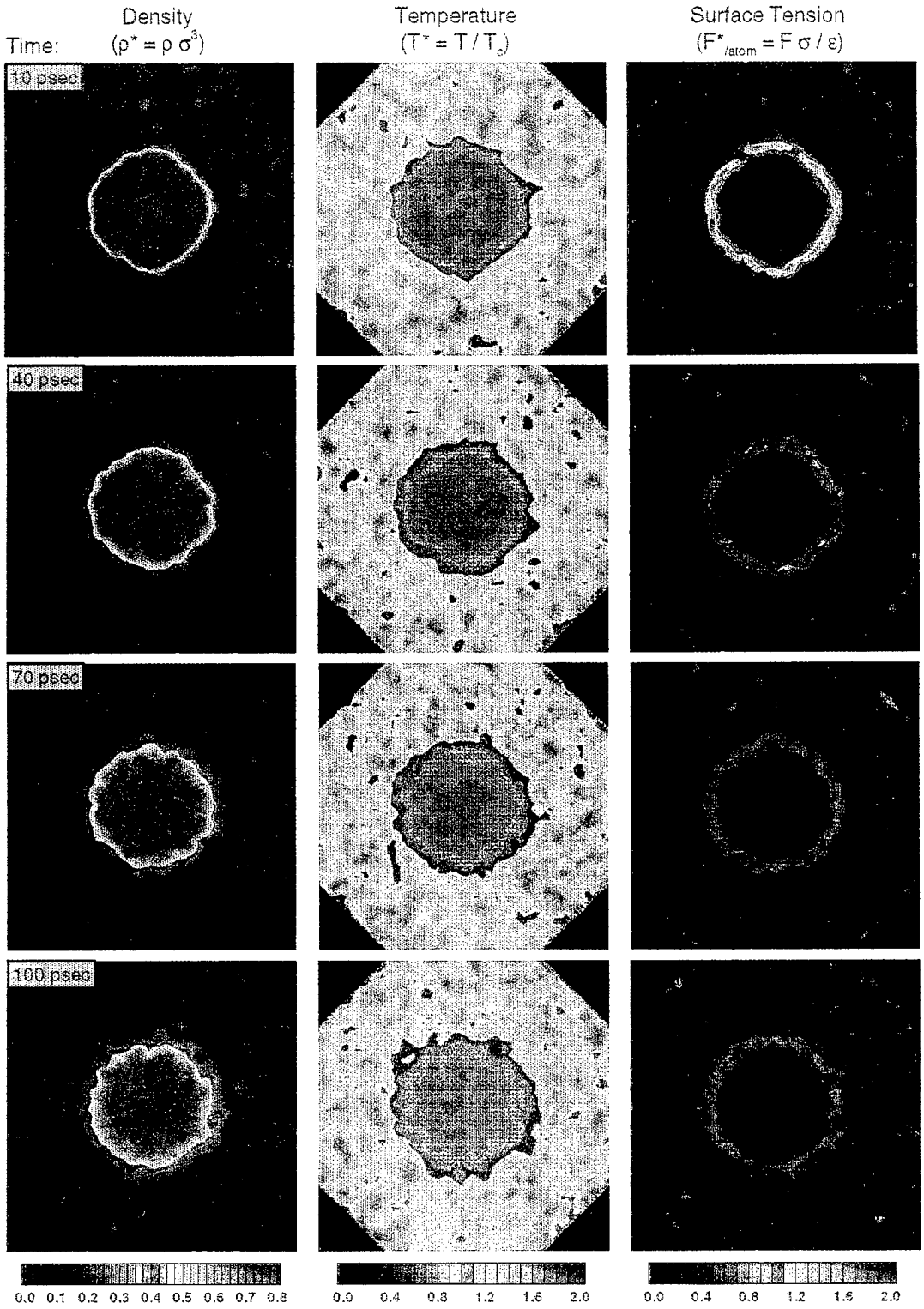


Figure 5.24. Scaling level 2: Initial images. A 100,570 atom drop equilibrated to 100 K and 1200 kg/m^3 is placed into a 200 K, 7.5 MPa, and 236 kg/m^3 supercritical environment.

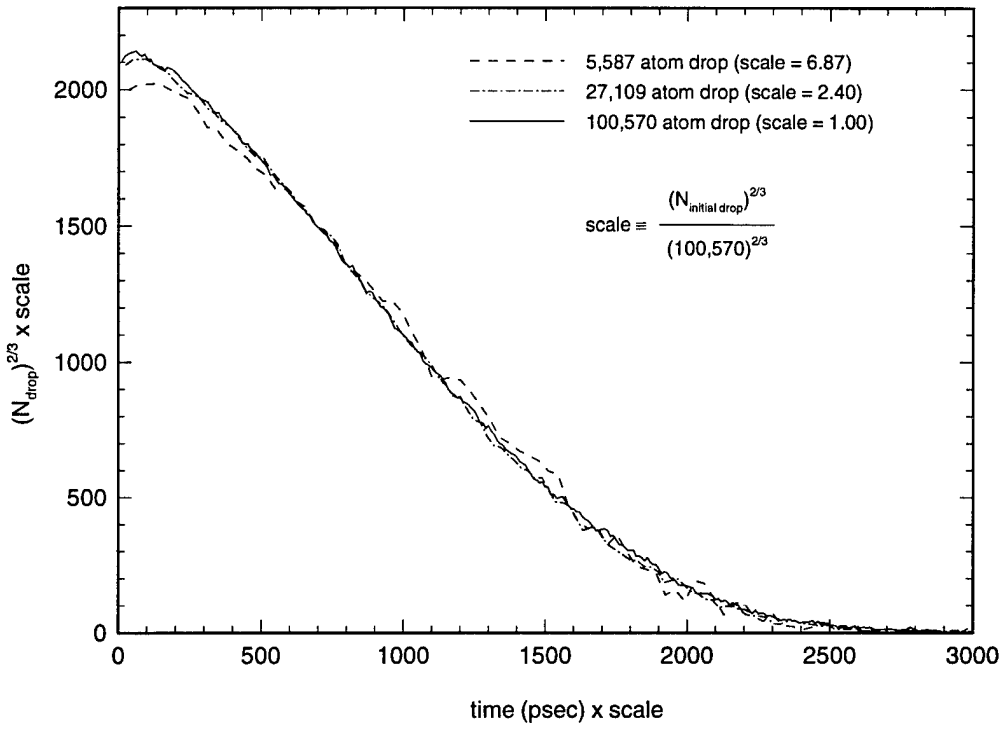


Figure 5.25. Scaled $N^{\frac{2}{3}}$ regression plots. Scales based on initial droplet sizes as shown.

drop can be ignored as a scaling parameter. (Volume effects would counter the purely surface scaling assumptions.) They show the droplet interior quickly reaches steady thermal and density conditions. In other words, the heating of the droplet volume is rapid compared to the surface diffusion rate. The volume therefore can be considered negligible as a scaling parameter.

The success of both simulating and scaling the diffusion process in the supercritical environment provides strong incentives to continue to pursue this modeling development. The next chapter provides a brief synopsis of the work detailed herein, and also details some of the anticipated future efforts.

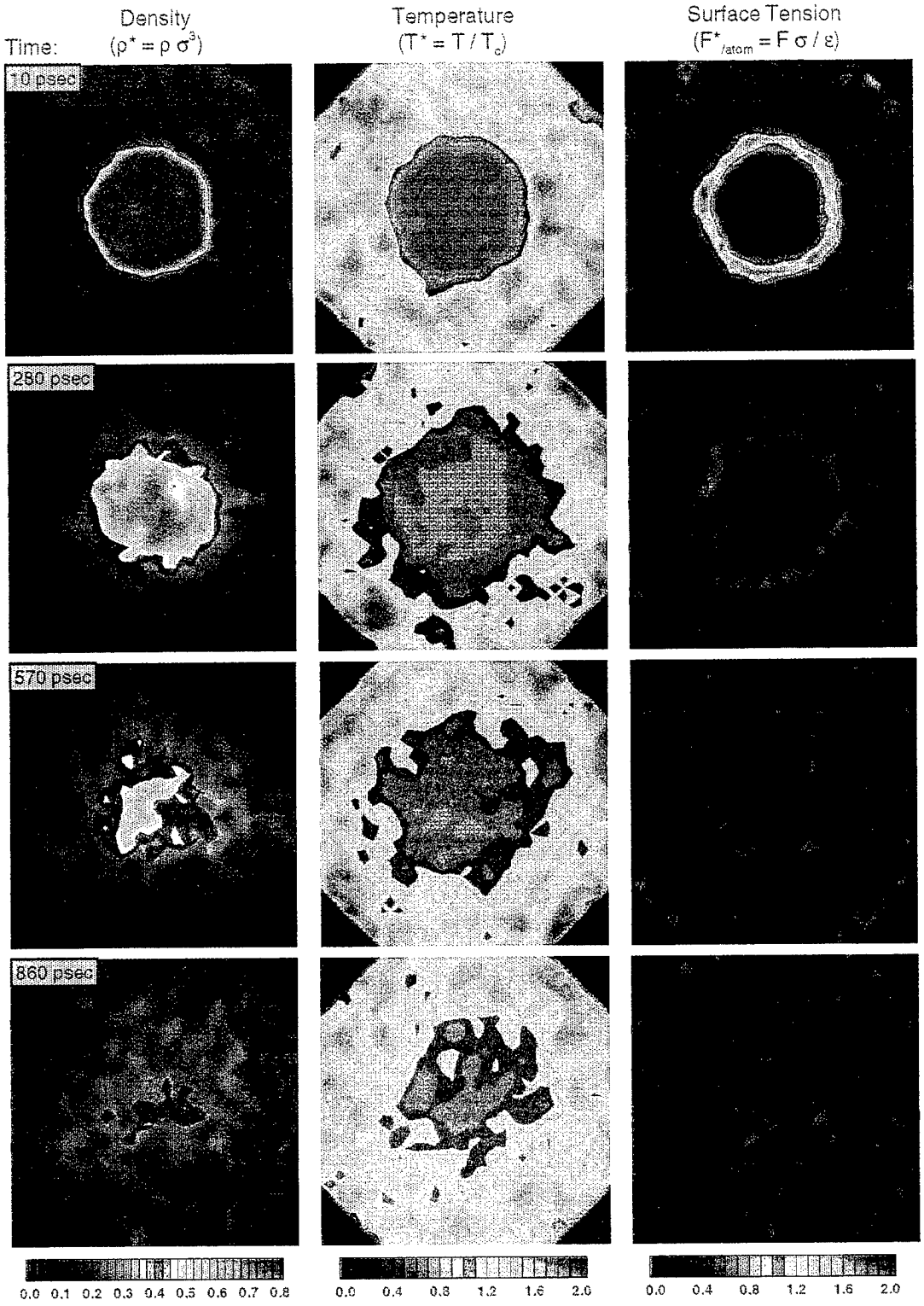


Figure 5.26. Scaling level 1: Full evaporation. A 27,109 atom drop equilibrated to 100 K and 1200 kg/m^3 is placed into a 200 K, 7.5 MPa, and 236 kg/m^3 supercritical environment.

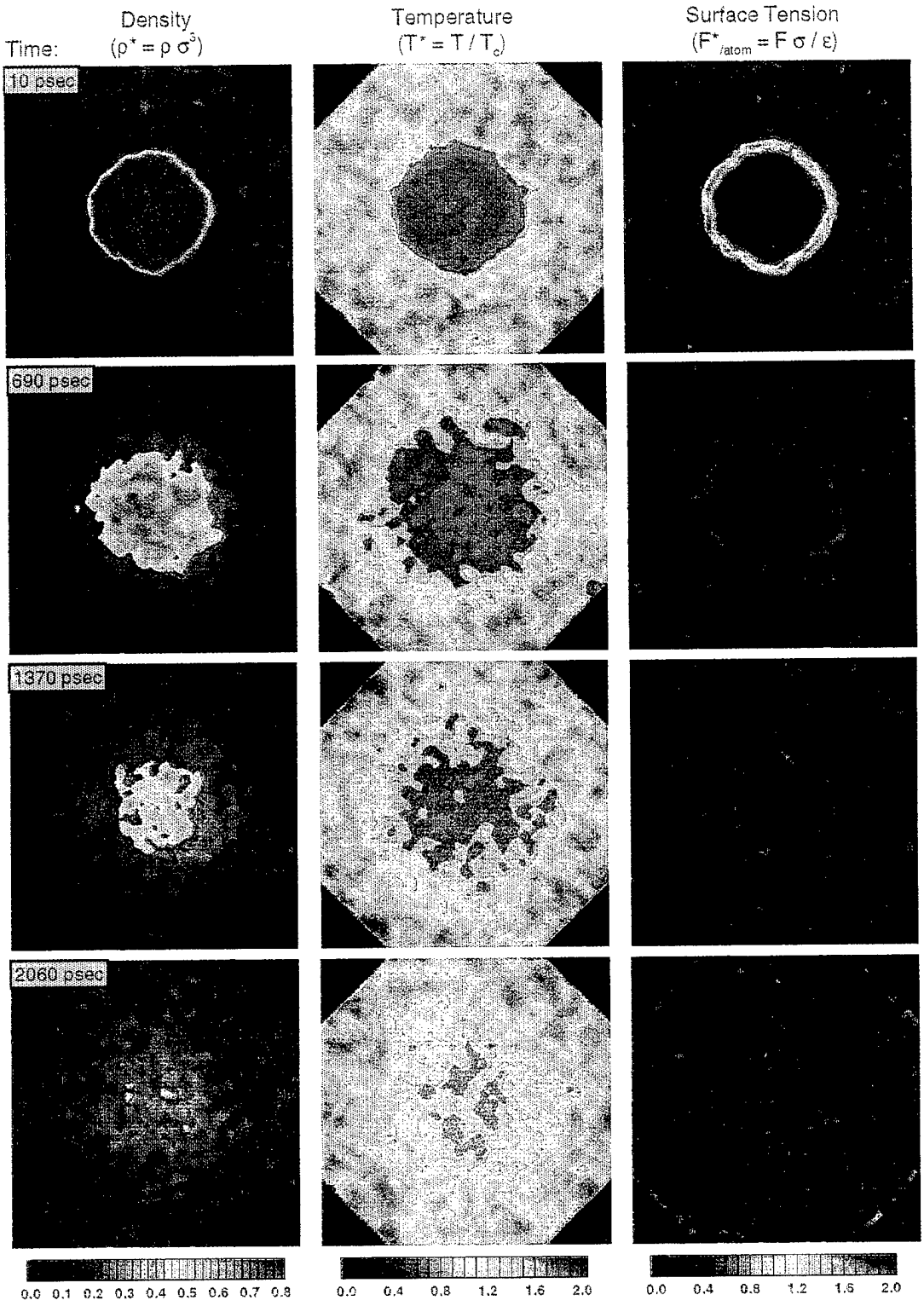


Figure 5.27. Scaling level 2: Full evaporation. A 100,570 atom drop equilibrated to 100 K and 1200 kg/m^3 is placed into a 200 K, 7.5 MPa, and 236 kg/m^3 supercritical environment.

Chapter 6

CONCLUSIONS

Combustion in a supercritical environment is a highly efficient, and therefore desirable, means of transferring fuel energy for rocket and turbojet propulsion. This thesis represents a first step toward the development of an evaluation tool to better understand this important process. The scope of the work is limited to evaluating a simple monatomic structure evaporating into high temperature and pressure surroundings. Real fuels are at least diatomic in structure and evaporate into complex and varied oxidizer environments. In addition to the associated requirement for more fine tuned inter-atomic potentials, the potentials between a large assortment of dissimilar elements are required. As the reader may note at this stage, the stepping from the evaporation model detailed in this thesis to an actual combustion process is a large step indeed. This chapter briefly reviews the successes of the work detailed herein and provides descriptions of other on-going associated work all performed ultimately to approach the goal of combustion simulations.

The response of an evaporating droplet in supercritical environments is a challenging process to evaluate. The lack of a saturated vapor condition, at which the droplet surface can reside, forces a near-critical point balance instead. The critical point for any substance is naturally unstable. Developing analytical models which include this region is therefore difficult. Molecular dynamic simulations, however, avoid many of the critical point pitfalls by modeling the atomic actions from the first principle of inter-atomic potentials. The work detailed herein showed that, indeed, a critical boundary could be simulated and droplet evaporation investigated. The discovery of a surface based diffusion was enlightening and allowed a theoretical scaling to very large systems. One point of caution, however, is warranted here. The entire simulation hinges on the inter-atomic potentials. This function provides the important first principles from which all the resulting physics evolve. If the potential is inaccurate, the simulation is not to be trusted.

The potential utilized in this study was simply a liquid-based, Lennard-Jones effective potential. Certainly the wide range of environments simulated during the study should be placed in question as a result. A project which is near completion [51] provides insight into the appropriate level of concern which should be applied.

The study is a duplication of the 6,000 atom droplet diffusion simulations using 3-body forces instead of the pair-wise effective forces. Multiple body potentials remove the necessity for adjustments due to density. In other words, they model the gas phases as accurately as the liquid with a single potential. Unfortunately, the 3-body force computations are significantly more complex and require a full order of magnitude greater processing time than the computations required for pairwise simulations. This author has reviewed the initial data from the 3-body simulations and encouraging results were seen. The high temperature, subcritical run (with an environment at 300 K and 3.0 MPa) differed the most. This would be expected since the gas phase in this environment is very dilute. The difference for this case, however, is only about 10% when comparing ${N_{drop}}^{2/3}$ regression rates. Also, the supercritical run showed only a slight shift. Apparently, the high density supercritical environments are simulated very well by the liquid based effective potential. Interestingly, the 3-body evaporation rate more closely matches the D^2 evaporation calculation. But the associated assumptions used in regards to property data cause the D^2 evaporation law rates to be as much in question as the simulation rates.

So the simple monatomic model investigated herein appears to have been a valid one. No fuels, however, exist as monatomic substances. In pursuit of real fuel evaporation insights a similar model is already in active use to investigate the evaporation of diatomic elements [52]. The initial results are encouraging in this area. The oxygen self diffusion appears to closely match the results from the argon study, at least qualitatively. The surface tension dissipates in a similar manner and a resultant scaling ability is expected to be found.

The simulation of an oxygen drop evaporating into a supercritical oxygen environment is a precursor, however, to the first 'real world' investigations. The model which simulates the oxygen evaporating into oxygen vapor will be adapted to simulate oxygen evaporating into hydrogen, a real fuel and oxidizer combination used in

many rocket motors. The major challenge here concerns the proper potentials not only for oxygen and for hydrogen, but also for the interactions between these two elements. Once again, the entire process hinges on the proper selection and use of these potentials.

Another related simulation in very early development stages is the modeling of injector flows. Many of the supercritical applications for future aerospace vehicles involve the heating of the fuel to supercritical temperatures and pressures before even interacting with the reacting environment. The concept is based on using the fuel as a coolant for the aircraft structures in transonic and potentially hypersonic environments. Such a highly energized flow should combust quite readily, but much of the actual physical understanding associated with this proposal is missing. Molecular dynamics promises to provide the insight desired for this problem in a similar manner to the insight derived in this thesis.

The property measures, performed on a linked cell basis, provided much of the insightful information in this thesis. They only included density, temperature and the qualitative surface tension, however. These properties are among the easiest to measure in molecular dynamics. Research is currently progressing to expand the cell-based property concept to the more detailed measures of transport coefficients [53]. The self-diffusivity, the shear viscosity and the thermal conductivity all can be transferred directly to continuum-based models to combine the first principle accuracy of molecular dynamics with the large scale capabilities of continuum analysis. With proper potentials, this work could significantly extend the validity of many standard droplet analysis techniques. Additionally, the MD analysis could allow the tabulation of thermodynamic and transport properties for complex molecules. The applications are then very broad-ranged.

In conclusion, the research performed in this thesis was productive and insightful. A parallel code for modeling droplet diffusion was developed which performs at an even level with some of the best developed to date. The investigations provided contour imaging of the highly energetic reactions to supercritical environments. Also, a verification of surface diffusion in these environments was made. The ability to scale the results helped prove the surface-based physics as well as provided validity

for the molecular dynamic code as an engineering tool. Future improvements in parallel architectures will only enhance the value of the code and molecular dynamics in general.

REFERENCES

[1] Kuo, K. K. *Principles of Combustion.* John Wiley and Sons, New York, NY, 1986.

[2] Vasserman, A. A., Kazavchinskii, Y. Z., and Rabinovich, V. A. *Thermophysical Properties of Air and Air Components.* Israel Program for Scientific Translations, Jerusalem, 1971.

[3] Reynolds, W. C. *Thermodynamic Properties in SI.* Department of Mechanical Engineering, Stanford University, Stanford, CA, 1979.

[4] Parker, S. P. *McGraw-Hill Dictionary of Scientific and Technical Terms, Fifth Edition.* McGraw–Hill, Inc., New York, NY, 1994.

[5] Levelt Sengers, J. M. H. *Critical Behavior of Fluids: Concepts and Applications.* Kluwer Academic Publishers, Netherlands, 1994.

[6] Tabor, D. *Gases, Liquids, Solids: and other states of matter.* Cambridge University Press, New York, NY, 1991.

[7] Cengel, Y. A., and Boles, M. A. *Thermodynamics: an Engineering Approach.* McGraw Hill, Inc., New York, NY, 1989.

[8] Cummings, P. T. *Molecular Simulation of Near–Critical and Supercritical Fluids.* Kluwer Academic Publishers, Netherlands, 1994.

[9] Burke, S. P., and Schumann, T. E. W. Diffusion flames. *Industrial and Engineering Chemistry Combustion Symposium,* **20**, pp. 998–1004, 1928.

[10] Spalding, D. Theory of particle combustion at high pressures. *ARS Journal,* **29**, p. 828, 1959.

[11] Rosner, D. On liquid droplet combustion at high pressures. *AIAA Journal,* **5**, p. 163, 1967.

[12] Savery, C. W., and Borman, G. L. Experiments on droplet vaporization at supercritical pressures. In *AIAA Paper No. 70–6, AIAA 8th Aerospace Sciences Meeting, New York.*, 1970.

[13] Manrique, J. A., and Borman, G. L. Calculations of steady state droplet vaporization at high ambient pressures. *Int. J. Heat and Mass Transfer*, **12**, pp. 1081–1095, 1969.

[14] Yang, V., Lin, N. N., and Shuen, J. S. Vaporization of liquid oxygen (LOX) droplets in supercritical hydrogen environments. *Combustion Sci. and Tech.*, **97**, pp. 247–270, 1994.

[15] Williams, F. A. The next 25 years of combustion theory. *Combustion Sci. and Tech.*, **98**, pp. 361–366, 1994.

[16] Smyth, K. C. Combustion metrology: A manifesto. *Combustion Sci. and Tech.*, **98**, pp. 341–347, 1994.

[17] Brezinsky, K. The next 25 years of combustion research: One researcher's perspective. *Combustion Sci. and Tech.*, **98**, pp. 237–243, 1994.

[18] Evans, D. J., and Hoover, W. G. Flows far from equilibrium via molecular dynamics. *Annual Rev. Fluid Mech.*, **18**, pp. 243–264, 1986.

[19] Beazley, D. M., and Lomdahl, P. S. Large–scale molecular dynamics on mpps. *SIAM News*, **28**, pp. 1–5, 1995.

[20] Sengers, J. V., and Levelt Sengers, J. M. H. Thermodynamic behavior of fluids near the critical point. *Annual Rev. Phys. Chem.*, **37**, pp. 189–222, 1986.

[21] Thompson, S. M., Gubbins, K. E., Walton, J. P. R. B., Chantry, R. A. R., and Rowlinson, J. S. A molecular dynamics study of liquid drops. *J. Chem. Phys.*, **81**, pp. 530–542, 1984.

[22] Maruyama, S. Surface phenomena of molecular clusters by molecular dynamics method. In *The Japan–U.S. Seminar on Molecular and Microscale Transport Phenomena (J–5).*, 1993.

[23] Matsumoto, M., Yasuoka, K., and Kataoka, Y. Microscopic features of evaporation and condensation at liquid surfaces: Molecular dynamics simulation. In *The Japan–U.S. Seminar on Molecular and Microscale Transport Phenomena (J–8).*, 1993.

[24] Ohara, T., and Aihara, T. Molecular dynamics study on structure of near–critical water. In *The Japan–U.S. Seminar on Molecular and Microscale Transport Phenomena (J–14).*, 1993.

[25] Koplick, J., and Banavar, J. R. Continuum deductions from molecular hydrodynamics. *Annual Rev. Fluid Mech.*, **27**, pp. 257–292, 1995.

[26] Hoover, W. G. *Nonequilibrium Molecular Dynamics at Livermore and Los Alamos.* Plenum Press, New York, NY, 1992.

[27] Sato, H., Iwama, Y., Kawakika, S., Saito, M., Morikami, K., Yao, T., and Tsutsumi, S. Parallelization of amber molecular dynamics program for the ap1000 highly parallel computer. IEEE 0–8186–2775–1/92, 1992.

[28] Brown, D., Clarke, J. H. R., Okuda, M., and Yamazaki, T. A molecular dynamics study of chain configurations in n–alkane–like liquids. *J. Chem. Phys.*, **100**, p. 6011, 1994.

[29] Brown, D., Clarke, J. H. R., Okuda, M., and Yamazaki, T. A domain decomposition parallelization strategy for molecular dynamics simulations on distributed memory machines. *Computer Physics Communications*, **74**, p. 67, 1993.

[30] Brown, D., Clarke, J. H. R., Okuda, M., and Yamazaki, T. A domain decomposition parallel processing algorithm for molecular dynamics simulations of polymers. *Computer Physics Communications*, **83**, pp. 1–13, 1994.

[31] Kalia, R. K., de Leeuw, S., Nakano, A., and Vashishta, P. Molecular dynamics simulations of coulombic systems on distributed–memory MIMD machines. *Computer Physics Communications*, **74**, pp. 316–326, 1993.

[32] Plimpton, S. Fast parallel algorithms for short–range molecular dynamics. *Journal of Computational Physics*, **117**, pp. 1–19, 1995.

[33] Reed, M. S. C., and Flurchick, K. M. Hybrid molecular dynamics: An approach to low density simulations. *Computer Physics Communications*, **81**, pp. 56–64, 1994.

[34] Maitland, G. C., and Smith, E. B. The intermolecular pair potential of argon. *Molecular Physics*, **22**, pp. 861–868, 1971.

[35] Haile, J. M. *Molecular Dynamics Simulation: Elementary Methods.* John Wiley and Sons, New York, NY, 1992.

[36] Allen, M. P., and Tildesley, D. J. *Computer Simulation of Liquids.* Oxford University Press, New York, NY, 1984.

[37] Verlet, L. Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Phys. Rev.*, **159**, pp. 98–103, 1967.

[38] Quentrec, B., and Brot, C. New method for searching for neighbors in molecular dynamics computations. *Journal of Computational Physics*, **13**, pp. 430–432, 1975.

[39] Larsen, B. Summary of session I. In *NRCC Proceedings No. 9, Menlo Park, California.*, 1980.

[40] Adams, D. J. Periodic, truncated–octahedral boundary conditions. In *NRCC Proceedings No. 9, Menlo Park, California.*, 1980.

[41] Pearlman, D. A. S., Case, D. A., Caldwell, J. W., Ross, W. S., Cheatham, T. E., Ferguson, D. M., Seibel, G. L., Singh, U. C., Weiner, P. K., and Kollman, P. A. AMBER 4.1. University of California, San Francisco, 1995.

[42] Gropp, W., Lusk, E., and Skjellum, A. *Using MPI: Portable Parallel Programming with the Message–Passing Interface.* The MIT Press, Cambridge, MA, 1994.

[43] Nicolas, J. J., Gubbins, K. E., Streett, W. B., and Tildesley, D. J. Equation of state for the lennard–jones fluid. *Molecular Physics*, **37**, pp. 1429–1454, 1979.

[44] Steele, W. Personal communication, 1995.

[45] Schofield, P. Computer simulation studies of the liquid state. *Computer Physics Communications*, **5**, pp. 17–23, 1973.

[46] Press, W. H., Teukolsky, S. A., Vetterling, W. H., and Flannery, B. P. *Numerical Recipes.* Cambridge University Press, New York, NY, 1992.

[47] Grest, G. S., Dunweg, B., and Kremer, K. Vectorized link cell fortran code for molecular dynamics simulations for a large number of particles. *Computer Physics Communications*, **55**, p. 269, 1989.

[48] Hirschfelder, J. O., Curtiss, C. F., and Bird, R. B. *Molecular Theory of Gases and Liquids.* John Wiley and Sons, New York, 1954.

[49] Bird, R. B., Stewart, W. E., and Lightfoot, E. N. *Transport Phenomena.* John Wiley and Sons, New York, 1960.

[50] Lide, D. R. *CRC Handbook of Chemistry and Physics.* CRC Press, Boca Raton, 1996.

[51] Ohlandt, C. J. Molecular dynamics simulation of argon droplet evaporation with three body force potentials. Master's thesis, Pennsylvania State University, 1996.

[52] Kaltz, T. L. *Projected work.* Ph.D. thesis, Pennsylvania State University, 1997.

[53] Nwobi, O. C. *Projected work.* Ph.D. thesis, Pennsylvania State University, 1997.

Appendix A

TRUNCATED OCTAHEDRON CELL MAP

This appendix gives the listing of the subroutine used to generate the cell maps for the truncated octahedron boundary code.

```
C
C*********************************************************************
C  *** Subroutine to set cell interaction structures ***
C*********************************************************************
C
      Subroutine cellgen(mc)
C
      include 'evapto.inc'
C
      integer mc2,msq2,mcq4,l,im3,im4,im5,im6,im7,im8,
     & imagehexp,disphexp,mface,mtotal,intc,
     & ilf,irf,ibf,itf,iff,iaf,iq,xsign,ysign,zsign,mcface,
     & lfcells(20000),rfcells(20000),bfcells(20000),
     & tfcells(20000),ffcells(20000),afcells(20000),
     & image1(100000),image2(100000),image3(100000),image4(100000),
     & image5(100000),image6(100000),image7(100000),image8(100000)
C
      logical iwrite
C
      common/celldat/vex0,vex1,vex2,
     &      lastin,imface,icperq,cnlast,lasthexp,
     &      lface1,lface2,rface1,rface2,bface1,bface2,
     &      tface1,tface2,fface1,fface2,aface1,aface2,
     &      quad,cellact,gocell,image
C
      rm = dble(mc)
      mc2 = 2*mc
      msq = mc*mc
      msq2 = 2*msq
      mcube = msq*mc
C
      if(mod(mc,4).eq.0) then
        mcq2 = mc/2
        mcq4 = mc/4
        lasthexp = mcq2 + mcq4
        imagehexp = lasthexp - 3
        disphexp = lasthexp - 2
        mface = (mc+4)*(mc+8)/8
        imface = (mc-8)*(mc-4)/8
        mtotal = mcube/2 + (3*msq)/2
        mcface = mc/2 - 1
        vex0 = (10.d0/3.d0)*dble((mcq2+mcq4)*(mcq2+mcq4+1)
     &          -3*mcq4*(mcq4+1))+dble(mc)
        vex1 = (2.d0/3.d0)*dble((mcq2+mcq4-1)*(mcq2+mcq4)
     &          -3*mcq4*(mcq4+1))
```

```
        vex2 = 0.d0
      else
        mcq2 = mc/2
        mcq4 = (mc+2)/4
        lasthexp = mcq2 + mcq4
        imagehexp = lasthexp - 4
        disphexp = lasthexp -2
        mface = (mc+6)*(mc+10)/8
        imface = (mc-6)*(mc-2)/8
        mtotal = mcube/2 + (9*msq)/4 - 5
        mcface = mc/2 - 1
        vex0 = (47.d0/12.d0)*dble((mcq2+mcq4)*(mcq2+mcq4+1)
     &          -3*mcq4*(mcq4+1))+dble(3*(mc+2))+.5d0*dble(mcq4-1)
        vex1 = 2.d0*dble((mcq2+mcq4-1)*(mcq2+mcq4)
     &          -3*mcq4*(mcq4+1))
        vex2 = (1.d0/12.d0)*dble((mcq2+mcq4-2)*(mcq2+mcq4-1)
     &          -3*(mcq4-1)*(mcq4))
      endif
C
      lface1 = mtotal - 6*mface + 1
      lface2 = lface1 + mface - 1
      rface1 = lface2 + 1
      rface2 = lface2 + mface
      bface1 = rface2 + 1
      bface2 = rface2 + mface
      tface1 = bface2 + 1
      tface2 = bface2 + mface
      fface1 = tface2 + 1
      fface2 = tface2 + mface
      aface1 = fface2 + 1
      aface2 = fface2 + mface
      cnlast = mtotal
C
      quad(1) = 1
      quad(2) = 1 + mc/2
      quad(3) = 1        + msq/2
      quad(4) = 1 + mc/2 + msq/2
      quad(5) = 1                + mcube/2
      quad(6) = 1 + mc/2         + mcube/2
      quad(7) = 1        + msq/2 + mcube/2
      quad(8) = 1 + mc/2 + msq/2 + mcube/2
C
C*********************************************************************
C     *** Setting neighbor cell assignment array ***
C*********************************************************************
C
C     --- interior cells ---
C
      ic = 1
      do 601 i=-1,1,1
        do 602 j=-mc,mc,mc
          do 603 k=-msq,msq,msq
            icell = i + j + k
            if(icell.eq.0) goto 603
            cellact(ic) = icell
            ic = ic + 1
  603   continue
  602  continue
  601 continue
```

```
C
C     --- left face ---
C
      do 541 i=-1,1,1
        do 542 j=-mc,mc,mc
          do 543 k=-msq,msq,msq
            icell = i + j + k
            if(icell.eq.0) goto 543
            if(i.eq.-1) icell = icell + mc
            cellact(ic) = icell
            ic = ic + 1
  543   continue
  542  continue
  541 continue
C
C     --- right face ---
C
      do 551 i=1,-1,-1
        do 552 j=-mc,mc,mc
          do 553 k=-msq,msq,msq
            icell = i + j + k
            if(icell.eq.0) goto 553
            if(i.eq.1) icell = icell - mc
            cellact(ic) = icell
            ic = ic + 1
  553   continue
  552  continue
  551 continue
C
C     --- bottom face ---
C
      do 561 j=-mc,mc,mc
        do 562 i=-1,1,1
          do 563 k=-msq,msq,msq
            icell = i + j + k
            if(icell.eq.0) goto 563
            if(j.eq.-mc) icell = icell + msq
            cellact(ic) = icell
            ic = ic + 1
  563   continue
  562  continue
  561 continue
C
C     --- top face ---
C
      do 571 j=mc,-mc,-mc
        do 572 i=-1,1,1
          do 573 k=-msq,msq,msq
            icell = i + j + k
            if(icell.eq.0) goto 573
            if(j.eq.mc) icell = icell - msq
            cellact(ic) = icell
            ic = ic + 1
  573   continue
  572  continue
  571 continue
C
C     --- fore face ---
C
```

```
      do 581 k=-msq,msq,msq
        do 582 i=-1,1,1
          do 583 j=-mc,mc,mc
            icell = i + j + k
            if(icell.eq.0) goto 583
            if(k.eq.-msq) icell = icell + mcube
            cellact(ic) = icell
            ic = ic + 1
  583   continue
  582  continue
  581 continue
C
C     --- aft face ---
C
      do 591 k=msq,-msq,-msq
        do 592 i=-1,1,1
          do 593 j=-mc,mc,mc
            icell = i + j + k
            if(icell.eq.0) goto 593
            if(k.eq.msq) icell = icell - mcube
            cellact(ic) = icell
            ic = ic + 1
  593   continue
  592  continue
  591 continue
C
C*********************************************************************
C     *** Setting cell loop order array ***
C*********************************************************************
C
C     --- order is from inside out along hexagon planes ---
C
      iwrite = .false.
C
      do 500 m=1,lasthexp
        if(m.eq.imagehexp) iwrite = .true.
        if(m.eq.disphexp) lastin = intc
        n = m-1
        do 501 iq=1,8
          xsign = 1
          ysign = 1
          zsign = 1
          if((iq.eq.1).or.(iq.eq.3).or.
     &          (iq.eq.5).or.(iq.eq.7)) xsign = -1
          if((iq.eq.1).or.(iq.eq.2).or.
     &          (iq.eq.5).or.(iq.eq.6)) ysign = -1
          if((iq.eq.1).or.(iq.eq.2).or.
     &          (iq.eq.3).or.(iq.eq.4)) zsign = -1
          do 502 j=0,n
            l = n-j
            do 503 i=0,l
              k = l-i
              if((i.gt.mcface).or.(j.gt.mcface).or.
     &             (k.gt.mcface)) goto 503
              ic = i
              jc = j
              kc = k
              if(xsign.lt.0) ic = mcface-i
              if(ysign.lt.0) jc = mcface-j
```

```
                if(zsign.lt.0) kc = mcface-k
C
                icell = quad(iq) + ic + jc*mc + kc*msq
                  if(iwrite) then
                    if(iq.eq.1) then
                      im1 = im1 + 1
                      image1(im1) = icell
                    elseif(iq.eq.2) then
                      im2 = im2 + 1
                      image2(im2) = icell
                    elseif(iq.eq.3) then
                      im3 = im3 + 1
                      image3(im3) = icell
                    elseif(iq.eq.4) then
                      im4 = im4 + 1
                      image4(im4) = icell
                    elseif(iq.eq.5) then
                      im5 = im5 + 1
                      image5(im5) = icell
                    elseif(iq.eq.6) then
                      im6 = im6 + 1
                      image6(im6) = icell
                    elseif(iq.eq.7) then
                      im7 = im7 + 1
                      image7(im7) = icell
                    elseif(iq.eq.8) then
                      im8 = im8 + 1
                      image8(im8) = icell
                    endif
                  endif
C
                if(i.eq.mcface) then
                  if(xsign.lt.0) then
                    ilf = ilf+1
                    lfcells(ilf) = icell
                  else
                    irf = irf+1
                    rfcells(irf) = icell
                  endif
                  goto 503
                endif
                if(j.eq.mcface) then
                  if(ysign.lt.0) then
                    ibf = ibf+1
                    bfcells(ibf) = icell
                  else
                    itf = itf+1
                    tfcells(itf) = icell
                  endif
                  goto 503
                endif
                if(k.eq.mcface) then
                  if(zsign.lt.0) then
                    iff = iff+1
                    ffcells(iff) = icell
                  else
                    iaf = iaf+1
                    afcells(iaf) = icell
                  endif
```

```
              goto 503
            endif
C
            intc = intc+1
            gocell(intc) = icell
C
  503       continue
  502     continue
  501   continue
  500 continue
C
      do 504 j=1,6
        do 505 i=1,mface
          intc = intc + 1
          if(j.eq.1) gocell(intc) = lfcells(i)
          if(j.eq.2) gocell(intc) = rfcells(i)
          if(j.eq.3) gocell(intc) = bfcells(i)
          if(j.eq.4) gocell(intc) = tfcells(i)
          if(j.eq.5) gocell(intc) = ffcells(i)
          if(j.eq.6) gocell(intc) = afcells(i)
  505   continue
  504 continue
C
      icperq = im1
C
      do 506 iq=1,8
        do 507 i=1,im1
          if(iq.eq.1) then
            im = im+1
            image(im) = image1(i)
          elseif(iq.eq.2) then
            im = im+1
            image(im) = image2(i)
          elseif(iq.eq.3) then
            im = im+1
            image(im) = image3(i)
          elseif(iq.eq.4) then
            im = im+1
            image(im) = image4(i)
          elseif(iq.eq.5) then
            im = im+1
            image(im) = image5(i)
          elseif(iq.eq.6) then
            im = im+1
            image(im) = image6(i)
          elseif(iq.eq.7) then
            im = im+1
            image(im) = image7(i)
          elseif(iq.eq.8) then
            im = im+1
            image(im) = image8(i)
          endif
  507   continue
  506 continue
C
      return
      end
```

Appendix B

FUSE CODE

This appendix is a listing of the code used to fuse an equilibrated droplet into an environment. There is an option to generate a binary data set for visualization using Tecplot. Utilizing this capability requires access to Tecplot libraries. The fusing portion of the code can be used independent of the Tecplot visualization.

```
C**********************************************************************
C FILE: fuse.f
C DESCRIPTION:
C         Code to fuse equilibrated liquid and gas models.
C AUTHOR: Jeff Little
C LAST REVISED: 02/13/96
C**********************************************************************
      program fuse
      implicit none
C
      real*8    vx(500000),vy(500000),vz(500000)
      real*8    rx(500000),ry(500000),rz(500000)
      real*8    x,y,z,u,v,w,rhodrp,rhoenv,drpboxd,envboxd,sigsqf
      real*8    sigma,sigsq,fcut,rcut,rhocut,rsqmax,rsqmin,rcut2
      real*8    rsq,rmboxinv,rmq2,nnbor,rjisq,voutlyinv,rhocuti
      real*8    rhocutq2,rcutsq,rsmax,rsmin,xji,yji,zji,rjimag,pi
      real*8    ridjiqm,coscut,cellqs,rm,envdincr,rho,rimag,rcut2sq
C
      real*4    rhoi,boxinv,ivinv
      real*4    xtec(500000),ytec(500000),ztec(500000),ttec(500000)
      real*4    type(500000),size(500000)
C
      integer   i,ii,j,k,ndrp,nenv,na,nerase,egap,nodebal,nd,ndc,ne
      integer   mc,icell,mcube,m,ic,jc,kc,nbor,jo,msq,nec,ntec
      integer   head(300000),list(500000)
      integer   Debug,TecIni,TecZne,TecDat,TecEnd
C
      logical   nodat
C
      character*1 answer,NULLCHR
      character*32 maintitle,zonetitle
C
      parameter (cellqs = 2.5d0)
      parameter (sigma = 3.405d-10)
      parameter (sigsq = sigma**2)
      parameter (rcut = 2.5d0*sigma)
      parameter (pi = 3.14159265359d0)
C
      rcut2 = 2.d0*rcut
      rcut2sq = rcut2*rcut2
      voutlyinv = 1.d0/((4.d0/3.d0)*pi*((2.d0*rcut)**3))
```

```
      ivinv = 1.0/(0.5*sngl((4.d0/3.d0)*pi*(2.5d0*sigma)**3))
      sigsqf = (2.d0**(1.d0/3.d0))*sigsq
      fcut = sqrt(sigsqf)
      rcutsq = rcut*rcut
      write(*,*)
      write(*,*) 'fcut/sigma =',fcut/sigma
      NULLCHR = CHAR(0)
      Debug = 1
C
      OPEN(9,file='fuse.out',status='unknown',form='formatted')
      OPEN(11,file='r.evp',status='unknown',form='unformatted')
      OPEN(12,file='v.evp',status='unknown',form='unformatted')
      OPEN(14,file='r.drp',status='old',form='unformatted')
      OPEN(15,file='v.drp',status='old',form='unformatted')
      OPEN(16,file='r.env',status='old',form='unformatted')
      OPEN(17,file='v.env',status='old',form='unformatted')
C
C**********************************************************************
C  *** Setting fuse parameters ***
C**********************************************************************
C
      write(9,*)
      write(9,*) 'Fusing of droplet into surrounding environment'
      write(*,*)
      write(*,*) 'Fusing of droplet into surrounding environment'
      read(14) ndrp,rhodrp,drpboxd
      read(16) nenv,rhoenv,envboxd
C
      rm = envboxd/(cellqs*sigma)
      mc = int(rm)
C
      write(9,*)
      write(9,*) 'rhodrp =',real(rhodrp)
      write(9,*) 'rhoenv =',real(rhoenv)
      write(*,*)
      write(*,*) 'rhodrp =',real(rhodrp)
      write(*,*) 'rhoenv =',real(rhoenv)
      write(*,*)
      write(*,*) 'Input the density cutoff value:'
      write(*,*) '   (enter 0 to use the environment density)'
      read(*,*) x
      write(*,*)
      write(*,*) '...and input the view angle for environment checks:'
      read(*,*) coscut
      coscut = cos((coscut/360.d0)*pi)
      write(*,*) 'cos(coscut) = ',coscut
C
      if(x.eq.0) then
        rhocut = rhoenv
      else
        rhocut = x
      endif
C
      write(*,*)
      write(*,*) 'Enter rhocuti to define inner surface radius:'
      read(*,*) y
      rhocuti = y
      write(9,*)
      write(9,*) 'rhocut = ',sngl(rhocut)
```

```
      write(9,*) 'rhocuti = ',sngl(rhocuti)
      write(*,*)
      write(*,*) 'Enter the task balance value:'
      read(*,*) nodebal
      write(*,*)
      write(*,*) 'Do you wish to save the data (y/n)?'
      read(*,*) answer
        nodat = .true.
        if(answer.eq.'y') nodat = .false.
      write(9,*)
      write(9,*) 'task balance # = ',nodebal
      write(9,*)
      write(9,*) 'environment mc = ',mc,' (rm = ',sngl(rm),')'
C
C*********************************************************************
C  *** Erase liquid atoms in the vapor region based on rhocut ***
C*********************************************************************
C
      rsqmax = 0.d0
      rsqmin = 1.d10
      k = 0
      ndc = 0
      do 111 i=1,ndrp
        read(14) x,y,z,rhoi
          rhoi = rhoi*ivinv
c         if(mod(i,100).eq.0) write(*,*) 'rhoi(',i,') =',rhoi
        read(15) u,v,w
        if(rhoi.le.rhocut) then
          ndc = ndc + 1
          xtec(ndc) = sngl(x)
          ytec(ndc) = sngl(y)
          ztec(ndc) = sngl(z)
          ttec(ndc) = 5.0
          goto 111
        endif
        rsq = x**2 + y**2 + z**2
C        if(rsq.gt.rsqmax) rsqmax = r
C   ...rsqmax determined after throwing out outlyers (see below)
            k = k+1
            rx(k) = x
            ry(k) = y
            rz(k) = z
            vx(k) = u
            vy(k) = v
            vz(k) = w
            type(k) = 0.0
        if(rhoi.gt.rhocuti) goto 111
            type(k) = 1.0
            if(rsq.lt.rsqmin) rsqmin = rsq
  111 continue
C
      if(rsqmin.gt.(1.0d9)) rsqmin = 0.d0
C
C      --- droplet linked list ---
C
      rmboxinv = rm/envboxd
      rmq2 = rm/2.d0
      msq = mc*mc
      mcube = msq*mc
```

```
C
      do 97 icell = 1,mcube
        head(icell) = 0
  97  continue
      do 98 i=1,k
        icell = 1 + int(rx(i)*rmboxinv+rmq2)
     &            + int(ry(i)*rmboxinv+rmq2)*mc
     &            + int(rz(i)*rmboxinv+rmq2)*msq
        list(i) = head(icell)
        head(icell) = i
  98  continue
C
C      --- erase droplet outlyers ---
C
      jo = 0
      rhocutq2 = 0.5d0*rhocut
C
      do 201 m=1,mcube
      i = head(m)
  211 if(i.eq.0) goto 201
        x = rx(i)
        y = ry(i)
        z = rz(i)
        rsq = x*x + y*y + z*z
        if(rsq.lt.rsqmin) goto 224
        nnbor = 1.0
      do 221 ic = -2,2,1
        do 222 jc = -2,2,1
          do 223 kc = -2,2,1
            nbor = m + ic + jc*mc + kc*msq
              j = head(nbor)
  231         if(j.eq.0) goto 223
              if(j.ne.i) then
                rjisq = (rx(j)-x)**2 + (ry(j)-y)**2 + (rz(j)-z)**2
                if(rjisq.lt.rcut2sq) nnbor = nnbor + 1.0
              endif
              j = list(j)
              goto 231
  223     continue
  222   continue
  221 continue
C
C      --- outlyers defined as any atom whose surrounding density
C             (within twice the cutoff radius) is less than 0.5*rhocut ---
C
      rho = nnbor*voutlyinv
      if(rho.le.rhocut) then
        jo = jo+1
        ndc = ndc + 1
        xtec(ndc) = sngl(rx(i))
        ytec(ndc) = sngl(ry(i))
        ztec(ndc) = sngl(rz(i))
        ttec(ndc) = 3.0
        rx(i) = envboxd + 1.0
C        ...tagged as an outlyer
      endif
  224 i = list(i)
      goto 211
  201 continue
```

```
C
      j = 0
      do 301 i=1,k
        if(rx(i).gt.envboxd) goto 301
        j = j+1
        rx(j) = rx(i)
        ry(j) = ry(i)
        rz(j) = rz(i)
        vx(j) = vx(i)
        vy(j) = vy(i)
        vz(j) = vz(i)
        ntec = ndc + j
        xtec(ntec) = sngl(rx(i))
        ytec(ntec) = sngl(ry(i))
        ztec(ntec) = sngl(rz(i))
        ttec(ntec) = type(i)
        rsq = rx(j)**2 + ry(j)**2 + rz(j)**2
        if(rsq.gt.rsqmax) rsqmax = rsq
  301 continue
C
      nd = j
C
      rsmax = sqrt(rsqmax)
      rsmin = sqrt(rsqmin)
C
      write(9,*)
      write(9,*) 'ndrp in =',ndrp
      write(9,*) '  ndrp rhocutted =',ndrp-k
      write(9,*) '  ndrp outlyers  =',jo
      write(9,*) 'ndrp out =',j
      write(9,*)
      write(9,*) 'surface radius = (',sngl(rsmin/sigma),' to ',
     &           sngl(rsmax/sigma),')*sigma'
C
      ndrp = j
      rsqmax = (rsmax + fcut)**2
C
C      --- update droplet linked list ---
C
      do 497 icell = 1,mcube
        head(icell) = 0
 497  continue
      do 498 i=1,ndrp
        icell = 1 + int(rx(i)*rmboxinv+rmq2)
     &            + int(ry(i)*rmboxinv+rmq2)*mc
     &            + int(rz(i)*rmboxinv+rmq2)*msq
        list(i) = head(icell)
        head(icell) = i
 498  continue
C
C   --- filter out environment atoms within droplet volume ---
C
      k = ndrp
      nec = 0
      ne = 0
      do 501 i=1,nenv
C
        read(16) x,y,z,rhoi
        read(17) u,v,w
```

```
C
          rsq = x*x + y*y + z*z
          rimag = sqrt(rsq)
          if(rsq.lt.rsqmin) goto 503
          if(rsq.gt.rsqmax) goto 502
C
        icell = 1 + int(x*rmboxinv+rmq2)
     &            + int(y*rmboxinv+rmq2)*mc
     &            + int(z*rmboxinv+rmq2)*msq
C
       do 521 ic = -2,2,1
         do 522 jc = -2,2,1
           do 523 kc = -2,2,1
             nbor = icell + ic + jc*mc + kc*msq
               j = head(nbor)
  531          if(j.eq.0) goto 523
                 xji = rx(j)-x
                 yji = ry(j)-y
                 zji = rz(j)-z
                 rjimag = sqrt(xji*xji + yji*yji + zji*zji)
               if(rjimag.le.fcut) goto 503
               if(rjimag.le.rcut2) then
                 ridjiqm = ((x*xji)+(y*yji)+(z*zji))/(rjimag*rimag)
                 if(ridjiqm.ge.coscut) goto 503
               endif
               j = list(j)
               goto 531
  523      continue
  522    continue
  521  continue
C
C    --- if reach here, then save as environment atom ---
C
  502      k = k + 1
           rx(k) = x
           ry(k) = y
           rz(k) = z
           vx(k) = u
           vy(k) = v
           vz(k) = w
           goto 501
C
C    --- tecplot data of erased environment atoms ---
C
  503      nec = nec + 1
           ntec = ndc + nd + nec
           xtec(ntec) = sngl(x)
           ytec(ntec) = sngl(y)
           ztec(ntec) = sngl(z)
           ttec(ntec) = 7.0
           if(rsq.lt.rsqmin) ttec(ntec) = 6.0
  501  continue
C
       na = k
C
       write(9,*)
       write(9,*) 'nenv in =',nenv
C
       j = nenv - na + ndrp
```

```
      nenv = na - ndrp
C
      write(9,*) '  nenv erased =',j
      write(9,*) 'nenv saved =',nenv
C
      nerase = mod(na,nodebal)
      envdincr = (dble(na-nerase)/(0.5d0*(envboxd**3)))/rhoenv - 1.d0
C
      if(nerase.eq.0) then
        egap = nenv+1
      else
        egap = nenv/nerase
      endif
      k = 0
      m = 0
      do 631 j=1,nenv
        i = ndrp + j
        if(mod(j,egap).eq.0) then
          k = k + 1
          if(k.le.nerase) then
            ntec = ndc + nd + nec + k
            xtec(ntec) = sngl(rx(j+ndrp))
            ytec(ntec) = sngl(ry(j+ndrp))
            ztec(ntec) = sngl(rz(j+ndrp))
            ttec(ntec) = 8.0
            goto 631
          endif
        endif
        rx(i) = rx(j+ndrp)
        ry(i) = ry(j+ndrp)
        rz(i) = rz(j+ndrp)
        vx(i) = vx(j+ndrp)
        vy(i) = vy(j+ndrp)
        vz(i) = vz(j+ndrp)
        m = m+1
        ntec = nd + ndc + nec + nerase + m
        xtec(ntec) = sngl(rx(i))
        ytec(ntec) = sngl(ry(i))
        ztec(ntec) = sngl(rz(i))
        ttec(ntec) = 10.0
  631 continue
C
      nec = nec + k
      ne = m
C
      write(9,*)
      write(9,*) 'task balance erase =',nerase
      write(9,*) 'final nenv =',nenv-nerase
      write(9,*)
      write(9,*) 'final ntotal =',na-nerase
      write(9,*)
      write(9,*) 'Percent density increase due to evap:',
     &              sngl(envdincr*100.)
      write(9,*)
C
      if(nodat) then
        write(9,*) 'NOTE: data files not updated'
        write(9,*)
        goto 998
```

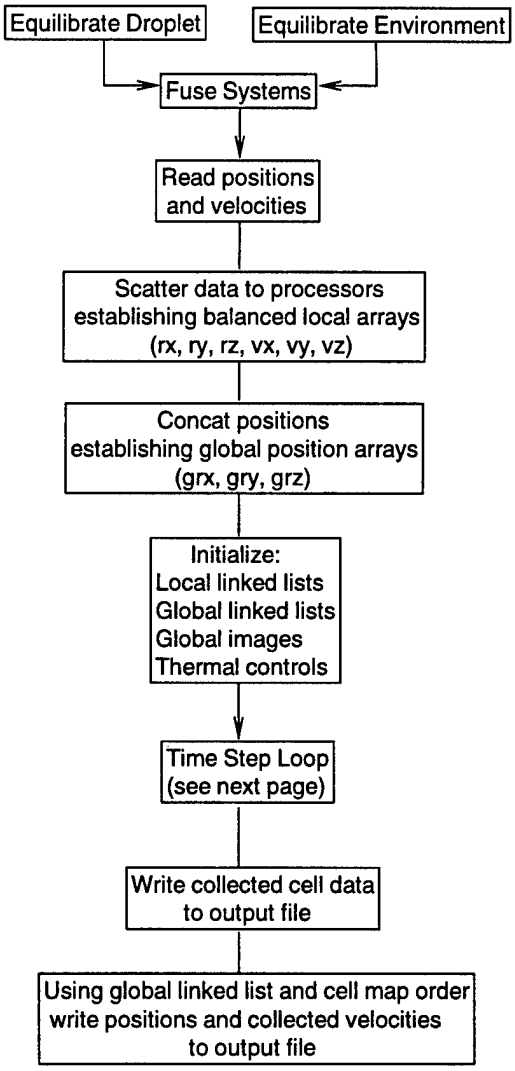
```
      else
        write(9,*) 'NOTE: data files updated'
        write(9,*)
        na = na - nerase
C
        rewind(11)
        write(11) na,envboxd
        do 401 i = 1,na
          write(11) rx(i),ry(i),rz(i)
          write(12) vx(i),vy(i),vz(i)
  401   continue
C
      endif
C
  998 continue
      write(*,*)
      write(*,*) 'Do you wish to view the data in Tecplot (y/n)?'
      read(*,*) answer
      if(answer.ne.'y') goto 999
C
      write(*,*)
      write(*,*) 'Enter the title for the tecplot dataset:'
      read(*,*) maintitle
C
C     ****** Initialize the Tecplot binary file ******
C
      I = TecIni(maintitle//NULLCHR,
     &           'x y z type size'//NULLCHR,
     &           'fuse.plt'//NULLCHR,
     &           '.'//NULLCHR,
     &           Debug)
C
C     ****** Zones: dropcut,outlyers,drop,envcut,nodebal,env
C
        boxinv = sngl(1.d0/envboxd)
C
        zonetitle = 'dropcut'
        ii = ndc-jo
        I = TecZne(zonetitle//NULLCHR,
     &             ii,1,1,
     &             'BLOCK'//NULLCHR)
        do 900 i=1,ii
          xtec(i) = xtec(i)*boxinv
          ytec(i) = ytec(i)*boxinv
          ztec(i) = ztec(i)*boxinv
          type(i) = ttec(i)
          size(i) = sigma*boxinv
  900   continue
        I = TecDat(ii,xtec,0)
        I = TecDat(ii,ytec,0)
        I = TecDat(ii,ztec,0)
        I = TecDat(ii,type,0)
        I = TecDat(ii,size,0)
C
        if(jo.ne.0) then
        zonetitle = 'outlyer'
        I = TecZne(zonetitle//NULLCHR,
     &             jo,1,1,
     &             'BLOCK'//NULLCHR)
```

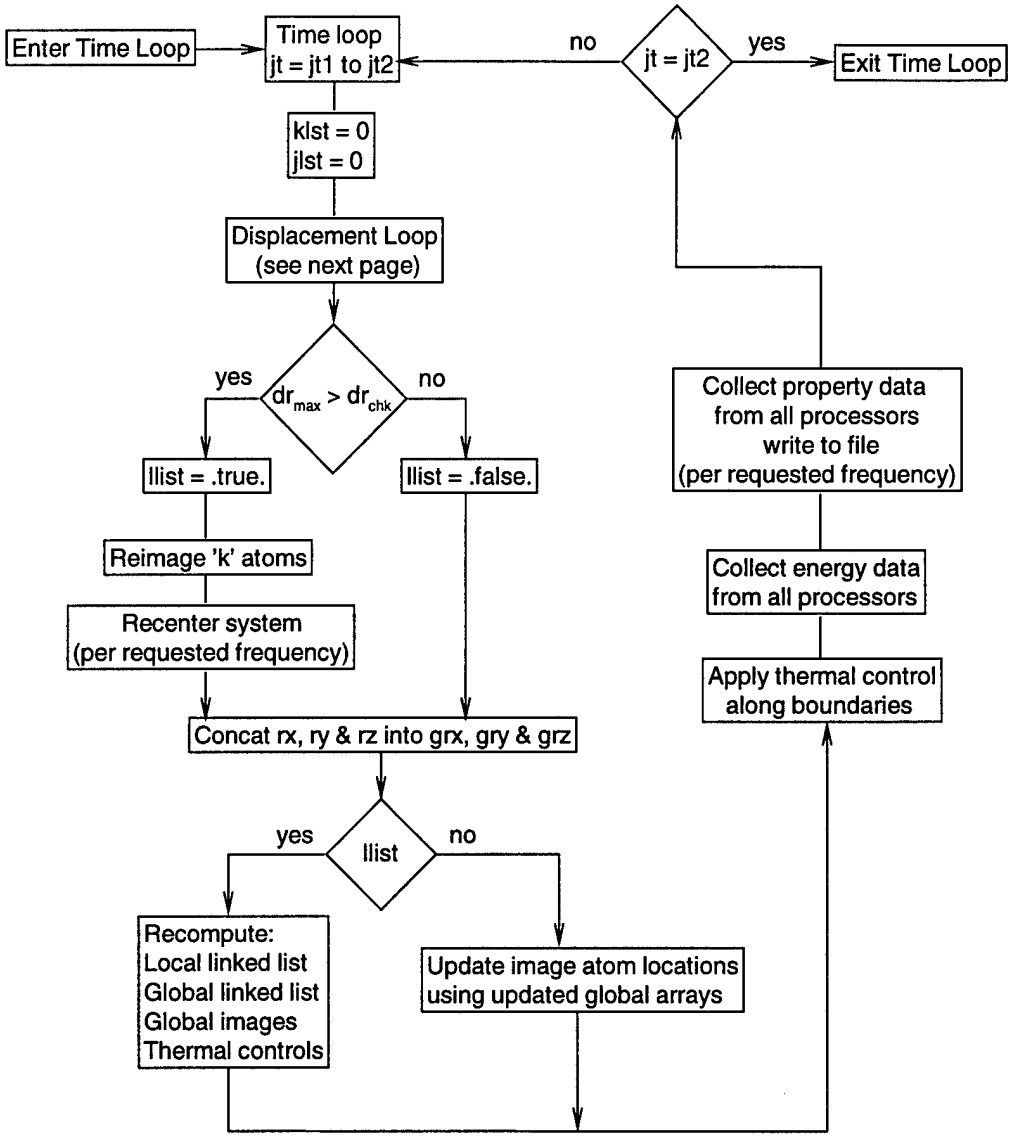
```
      do 901 i=1,jo
        xtec(i) = xtec(i+ii)*boxinv
        ytec(i) = ytec(i+ii)*boxinv
        ztec(i) = ztec(i+ii)*boxinv
        type(i) = ttec(i+ii)
        size(i) = sigma*boxinv
 901  continue
      I = TecDat(jo,xtec,0)
      I = TecDat(jo,ytec,0)
      I = TecDat(jo,ztec,0)
      I = TecDat(jo,type,0)
      I = TecDat(jo,size,0)
      endif
C
      zonetitle = 'drop'
      I = TecZne(zonetitle//NULLCHR,
     &           nd,1,1,
     &           'BLOCK'//NULLCHR)
      ntec = ndc
      do 902 i=1,nd
        xtec(i) = xtec(i+ntec)*boxinv
        ytec(i) = ytec(i+ntec)*boxinv
        ztec(i) = ztec(i+ntec)*boxinv
        type(i) = ttec(i+ntec)
        size(i) = sigma*boxinv
 902  continue
      I = TecDat(nd,xtec,0)
      I = TecDat(nd,ytec,0)
      I = TecDat(nd,ztec,0)
      I = TecDat(nd,type,0)
      I = TecDat(nd,size,0)
C
      zonetitle = 'envcut'
      ii = nec - nerase
      I = TecZne(zonetitle//NULLCHR,
     &           ii,1,1,
     &           'BLOCK'//NULLCHR)
C
      ntec = ndc + nd
      do 903 i=1,ii
        xtec(i) = xtec(i+ntec)*boxinv
        ytec(i) = ytec(i+ntec)*boxinv
        ztec(i) = ztec(i+ntec)*boxinv
        type(i) = ttec(i+ntec)
        size(i) = sigma*boxinv
 903  continue
      I = TecDat(ii,xtec,0)
      I = TecDat(ii,ytec,0)
      I = TecDat(ii,ztec,0)
      I = TecDat(ii,type,0)
      I = TecDat(ii,size,0)
C
      zonetitle = 'nodebal'
      I = TecZne(zonetitle//NULLCHR,
     &           nerase,1,1,
     &           'BLOCK'//NULLCHR)
C
      ntec = ndc + nd + nec - nerase
      do 904 i=1,nerase
```

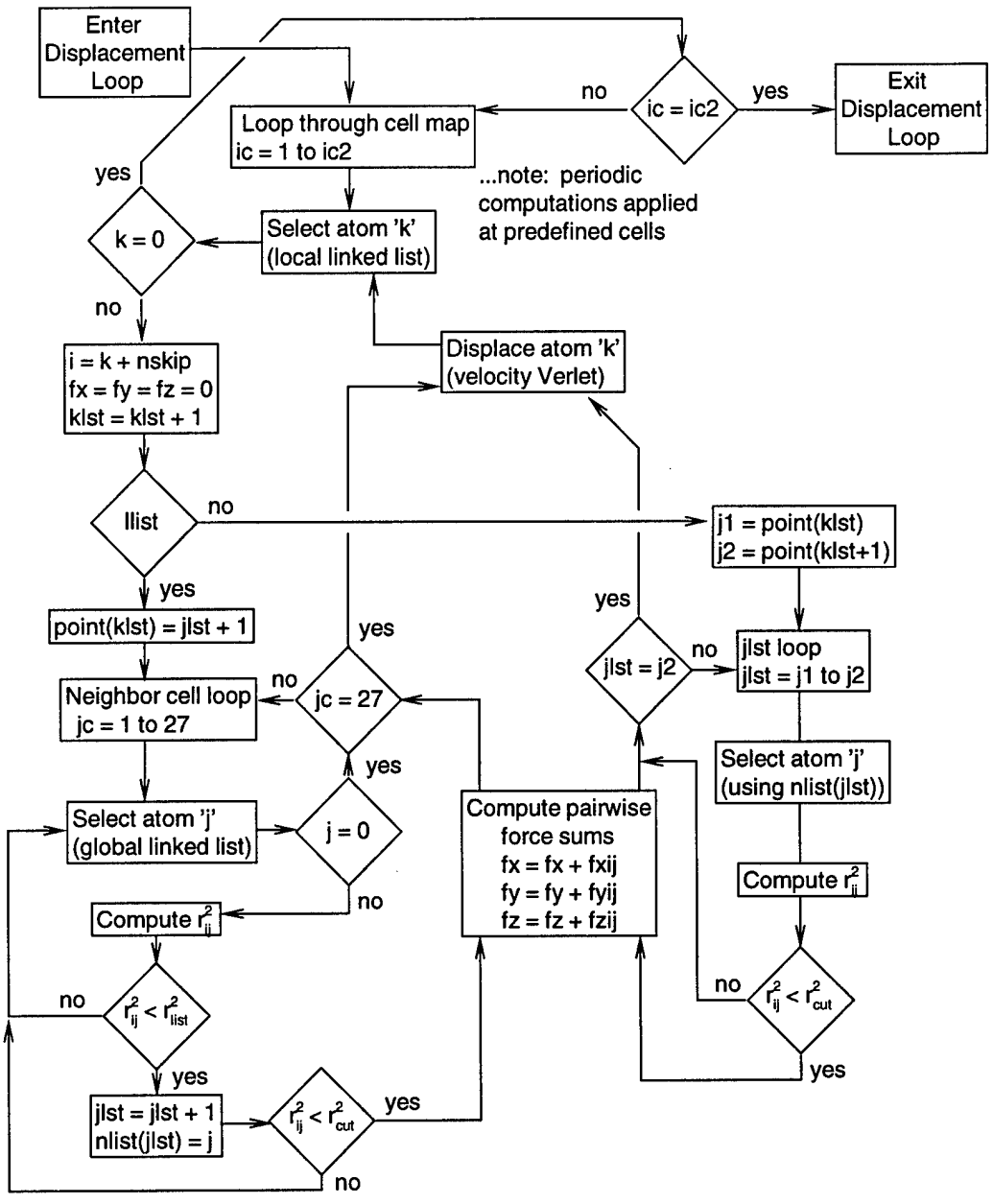
```
          xtec(i) = xtec(i+ntec)*boxinv
          ytec(i) = ytec(i+ntec)*boxinv
          ztec(i) = ztec(i+ntec)*boxinv
          type(i) = ttec(i+ntec)
          size(i) = sigma*boxinv
  904   continue
        I = TecDat(nerase,xtec,0)
        I = TecDat(nerase,ytec,0)
        I = TecDat(nerase,ztec,0)
        I = TecDat(nerase,type,0)
        I = TecDat(nerase,size,0)
C
        zonetitle = 'env'
        I = TecZne(zonetitle//NULLCHR,
     &             ne,1,1,
     &             'BLOCK'//NULLCHR)
        ntec = ndc + nd + nec
        do 905 i=1,ne
          xtec(i) = xtec(i+ntec)*boxinv
          ytec(i) = ytec(i+ntec)*boxinv
          ztec(i) = ztec(i+ntec)*boxinv
          type(i) = ttec(i+ntec)
          size(i) = sigma*boxinv
  905   continue
        I = TecDat(ne,xtec,0)
        I = TecDat(ne,ytec,0)
        I = TecDat(ne,ztec,0)
        I = TecDat(ne,type,0)
        I = TecDat(ne,size,0)
C
      I = TecEnd()
C
  999 continue
      stop
      end
```

Appendix C

EVAPORATION CODE FLOWCHART




Enter Time Loop
Time loop
jt = jt1 to jt2
no
jt = jt2
yes
Exit Time Loop
klst = 0
jlst = 0
Displacement Loop
(see next page)
yes
$dr_{max} > dr_{chk}$
no
llist = .true.
llist = .false.
Reimage 'k' atoms
Recenter system
(per requested frequency)
Concat rx, ry & rz into grx, gry & grz
yes
llist
no
Recompute:
Local linked list
Global linked list
Global images
Thermal controls
Update image atom locations
using updated global arrays
Collect property data
from all processors
write to file
(per requested frequency)
Collect energy data
from all processors
Apply thermal control
along boundaries


Enter Displacement Loop
Loop through cell map ic = 1 to ic2
ic = ic2
no
yes
Exit Displacement Loop
...note: periodic computations applied at predefined cells
yes
k = 0
Select atom 'k' (local linked list)
no
i = k + nskip
fx = fy = fz = 0
klst = klst + 1
Displace atom 'k' (velocity Verlet)
llist
no
yes
j1 = point(klst)
j2 = point(klst+1)
point(klst) = jlst + 1
yes
jlst = j2
no
jlst loop
jlst = j1 to j2
yes
jc = 27
no
Neighbor cell loop
jc = 1 to 27
Select atom 'j' (using nlist(jlst))
yes
j = 0
Compute pairwise force sums
fx = fx + fxij
fy = fy + fyij
fz = fz + fzij
Select atom 'j' (global linked list)
no
Compute r_{ij}^2
Compute r_{ij}^2
no
$r_{ij}^2 < r_{list}^2$
no
$r_{ij}^2 < r_{cut}^2$
yes
yes
jlst = jlst + 1
nlist(jlst) = j
$r_{ij}^2 < r_{cut}^2$
yes
no

VITA

Major Jeffery K. Little has served over fourteen years as an engineer and educator in the U.S. Air Force. Entering in 1982 after completing a Bachelor's of Science in Mechanical Engineering at Auburn University, he was assigned as a turbine engine research engineer at NASA Lewis Research Center. At the completion of this special duty assignment in 1984, he became an operations engineer for the Engine Test Facility at the Arnold Engineering Development Center (AEDC) in Tennessee. Over a four year tour he worked a variety of positions involving operations monitoring, test and resource scheduling, and facility modernization and maintenance. In addition, he earned a Master's of Science in Mechanical Engineering from the University of Tennessee Space Institute during his AEDC tour. In 1988 he was chosen as an AFROTC instructor and assigned to the University of Arizona in Tucson, Arizona. Here he taught history and was twice named the AFROTC Western Area instructor of the year. At the end of 1991, he was competitively chosen as an instructor by the Aeronautics Department at the USAF Academy in Colorado Springs, Colorado. He taught thermodynamics, heat transfer and energy conversion courses. In 1993 he was released early from the Academy assignment to pursue a sponsored doctorate in Aerospace Engineering at the Pennsylvania State University. This thesis represents the completion of this task. Major Little will follow this effort with an assignment at the sponsoring agency, the Aeronautics and Astronautics Department at the Air Force Institute of Technology.